A

SUN CITY
TI 99/4A
COMPUTER CLUB

PRESIDENT-	William Borchardt	755-1124
VICE-PRES-	William Brown	772-3921
TREASURER-	Charles Coy	751-1967
SECR/LIBR-	Dennis Hancock	855-1998
ASSIS. LIBR-	Mike Hunnicutt	543-2048

NEWSLETTER NO. 17 ** DATE Nov. 20,'86

NEXT MEETING: December 1, 1986 at the EL PASO TRADE SCHOOL 4710 ALABAMA 7pm.

For those that couldn't make the meeting, you missed a pretty productive evening. We did get elections off the ground, Dennis Hancock was voted back in to the Secretary/Librarian office. Nagy Zoltan declined a nomination to the office of Treasurer, and Charles Coy was subsequently elected in to the office. William Brown was elected in to the office of Vice-President. Yours truly was elected in to the office of President.

Charles Coy gave the Treasurers report. After the cost of the Monitor, and Newsletter mailing costs we will have approximately (we don't know the exact check charges) $210.00 in the Treasury.

Heino Huenken gave a demonstration of the Cor-Comp Triple Tech card. Showing us the features of the printer buffer, real time clock, and the speech synthesizer (which plugs in to the Triple Tech card). He followed that with a demonstration of the GRAM card from Mechatronic. Which, among its other features, allows the user to save modules to disk, load the disk to the GRAM card, and then execute the modules from the Master Title Screen without having to plug in the module.

This was the last meeting for Heino Heunken and Frank Huellen before they return to Germany. I'm sure I speak for everyone when I say it was nice to know Frank and Heino, and their presence at the meetings will surely be missed. On behalf of the club I wish them a safe journey back to Germany, and the best of luck in their endeavors in the years to come. Of course if either of you make it back here you will be welcome with open arms. Keep in touch with us.

MAILBAG

The big news in the mailbag this month is the receipt of the TI Diagnostic package. It was not without problems though, the postal service stuffed the 8"X11" envelope it came in inside our 4"X4" P.O. Box. Needless to say it was hard to load up the programs from bent disks. Through some timely effort I was able to get the programs off the disks though, and it looks real good. I don't have the Pascal system, and didn't build the interface cable for the RS232 tests, but I know that all other functions work, and assume that the Pascal and RS232 tests do also. I will have the disks, and hopefully, some more copies of the documentation at the meeting.

We also received an ad from Trinity Systems software for their Yak-Man program, the USA states and capitols game, and their Bible books game. From H and O enterprises we received an ad for Diskettes and supplies. I will have both of these ad's at the meeting.

In my mail box I finally received word from Great Lakes software on their Graphics program. Ernest Chandler from Great Lakes software tells me that indeed they will write print routines for those of us with odd ball printers (non Epson type) if we would send them a complete copy of our printer manuals. Including the control and graphics codes. So now there is no reason for anyone not to have a graphics program, including myself. The program is also now selling for $39.95 ten dollars cheaper than it was selling. He didn't mention any additional charges for this service either.

I have also received another letter from Dale Givens he sends his regards, and has sent this month's Strictly Basic article, and I must admit he did a much better job than what I have been doing.

>>>>STRICTLY TI BASIC<<<<

>>>READ<>DATA<>RESTORE<<<

>>>>>>>DALE GIVENS<<<<<<<

>>>>>>>>10 NOV 86<<<<<<<<

In a recent article we saw how information can be stored and retrieved form devices external to the CPU running the program. TI Basic also has provisions for storing and retrieving information within the program. This is especially useful if all you own is the basic console and a monitor. There are also occasions when small amounts of information need to be stored that don't justify external storage. Then there is my pet peeve the old * MEMORY FULL error. I'm sure this one has happened to everyone who did any programing prior to getting the PE Box and extra memory. With TI Basic, multiple statement lines are not allowed so a 'pipeline' structure of programing is imposed which uses more memory than multi-statement lines. After writing a game which would not fit in the 16K of the console I became an advocate of program compaction prior to purchasing Extended Basic. One of the ways I found to reduce the size of the program without changing the program was thru the use of DATA statements to store information used by other statements within the program. An example latter. Programer choice has a great deal to do with how a program handles information and often storing information within the program makes it more usable to others in the community who do not have all the extra peripherals. Three statements are used to access information from within the TI Basic program. They are RESTORE, READ and DATA.

DATA is used to store information within the program that can be used during program execution. Unlike using files, the programer must place this information in the program as it is written. The program can not store or put information in data statements, while it is running (arrays are used for this).

READ is used to get the information from the DATA statement so it can be used within the program. To further examine the READ and DATA statements compare it to the LET statement.

```
100 LET A=95
110 LET B$="FF818181818181FF"
120 LET R=10
130 LET C=15
140 CALL CHAR(A,B$)
150 CALL HCHAR(R,C,A)
160 GOTO 100      ! USE FCTN 4 TO STOP
```

In this program we tell the computer to make 'A' a numeric variable equal to 95, 'R' a numeric variable equal to 10 and 'C' a numeric variable equal to 15. We also define the string variable 'B$'. If the program is run a small box is placed on the screen at row 10, column 15. The equivalent program using the READ and DATA statements is:

```
100 READ A,B$
110 CALL CHAR(A,B$)
120 READ R,C
130 CALL HCHAR(R,C,A)
140 DATA 95;"FF818181818181FF",10,15
150 GOTO 130      ! USE FCTN 4 TO STOP
```

As this program runs; the first READ in line 100 searches for the DATA statement in line 140 and makes 'A' equal 95 and 'B$' equal the string. The second READ in line 120 again searches for the DATA statement in line 140 and makes 'R' equal 10 and 'C' equal 15.

To keep from getting confused the computer uses a pointer as it READs variables from the DATA statements. As each variable is READ the pointer is incremented so the next READ will get the next item of DATA. Notice in the second example that the GOTO statement does not jump to line 100 as prior example did. Change line 150 to GOTO 100 and run. You should get an * DATA ERROR IN 100 this time. Why? The internal pointer for the DATA statements was incremented four times prior to the GOTO statement thus when the program executed line 100 the second time it caused the READ statement to look for the fifth item in the DATA statements which doesn't exist; thus the error.

We can reset this DATA pointer by using the RESTORE statement. When the computer encounters the RESTORE by itself within the program it resets the pointer to the first DATA statement in the program. If the RESTORE is followed by a line number then the pointer is reset to that DATA statement wherever it is. The following program reads DATA from line 200, then 220, then 210 and then repeats the sequence.

```
100 READ A,B$,R,C,X
110 CALL CHAR(A,B$)
120 CALL HCHAR(R,C,A)
130 ON X GOTO 140,160,180
140 RESTORE
150 GOTO 100
160 RESTORE 210
170 GOTO 100      ! USE FCTN 4 TO STOP
180 RESTORE 220
190 GOTO 100
200 DATA 95,"FF818181818181FF",10,15,3
210 DATA 100,"FF81FFFF81FF81FF",15,10,1
220 DATA 42,"0F0909090909090F",12,12,2
```

Because the DATA statements contain the information the program will eventually use as a variable it can hold either numeric or string data or a combination. Within the DATA statement the comma is used to separate each variable. Care must be taken to insure that the correct variable type is available for the READ statement or errors will result. String data may be enclosed in quotes but it is not required unless the string contains a comma or leading or trailing spaces. If the string contains quotes then the quotes within the string must be double quotes.

```
100 READ A$,B$,C$,D$,E$,F$
110 PRINT A$
120 PRINT B$;C$
130 PRINT D$,E$
140 PRINT F$
150 DATA No quotes,"A comma, here"
160 DATA "    leading spaces"
170 DATA "Trailing spaces    ",there.
180 DATA """Quotes here!!"""
```

A null string or empty (same as LET A="") is set by adjacent commas. DATA statements can be located anywhere within the program. However at first I recommend that you put the DATA statements right after the program segment containing the READ statements to aid in debugging.

If you receive the error message * DATA ERROR IN nn, then one of two things are wrong in line nn. Either you have a RESTORE to a line number that doesn't exist or the variable in the DATA element does not match the READ variable. Another error that can result is * BAD VALUE IN nn. This can occur if the READ variable does not meet the parameters of the statement it is used in. This most commonly occurs when a READ statement for a string is equated to a numeric variable (numbers are valid strings thus no error in the READ).

Earlier I mentioned that the READ and DATA statements could be used to reduce the size of your program without changing it. Program One and Program Two below do exactly the same thing. Program One is 1831 bytes while Program Two is 1136 bytes. A savings of 695 bytes. However, you can't get something for nothing. Even though Program Two is smaller it takes longer to run. 10 seconds versus 8 seconds for Program One. Reason: the computer needs time to READ the DATA in Program Two.

Hope this helped you READ the DATA in the lines of strictly TI BASIC.

```
095 REM  PROGRAM ONE
100 CALL CLEAR
110 CALL SCREEN(3)
115 REM Define character patterns
116 REM for graphics
120 CALL CHAR(128,"FFFFFFFFFFFFFFFF")
130 CALL CHAR(129,"80808080808080E0")
140 CALL CHAR(130,"000000FFFFFFFF3F")
150 CALL CHAR(131,"FCFCFCFCFEFFFFFF")
160 CALL CHAR(132,"1F1F0F0703000000")
170 CALL CHAR(133,"FFFFFFBF3F3F3F1F")
180 CALL CHAR(134,"FFFFFFFEF8F0C000")
190 CALL CHAR(135,"0F07070301000000")
200 CALL CHAR(136,"FEFCFCF8F8FC3C18")
210 CALL CHAR(137,"00000000FF9F9FFF")
220 CALL CHAR(138,"00000000FFFFFFFF")
230 CALL CHAR(139,"CBDBCBDBC9FF0F0F")
240 CALL CHAR(140,"1155117575FFFFFF")
250 CALL CHAR(141,"117515D511FFFFFF")
260 CALL CHAR(142,"0402010000000000")
270 CALL CHAR(143,"0000000040201008")
275 REM Print LOGO info
280 PRINT TAB(10);"SUN CITY"::
290 PRINT TAB(10);"TI 99/4A"::
300 PRINT TAB(8);"COMPUTER CLUB"::::::::
::
305 REM Display graphics on screen
306 REM State of TEXAS
310 CALL HCHAR(11,5,137)
320 CALL HCHAR(11,6,138)
330 CALL HCHAR(11,7,138)
340 CALL HCHAR(12,5,139)
350 CALL HCHAR(12,6,140)
360 CALL HCHAR(12,7,141)
370 CALL HCHAR(13,5,142)
380 CALL HCHAR(13,6,143)
390 CALL HCHAR(14,6,142)
400 CALL HCHAR(14,7,143)
410 CALL HCHAR(15,7,142)
420 CALL HCHAR(15,8,143)
430 CALL HCHAR(16,8,142)
440 CALL HCHAR(16,9,143)
450 CALL HCHAR(17,9,142)
460 CALL HCHAR(17,10,143)
470 CALL HCHAR(18,10,142)
480 CALL HCHAR(18,11,143)
490 CALL HCHAR(19,11,142)
500 CALL HCHAR(19,12,143)
510 CALL HCHAR(20,12,142)
520 CALL HCHAR(20,13,143)
530 CALL HCHAR(21,13,142)
540 CALL HCHAR(20,15,128)
550 CALL HCHAR(20,16,129)
560 CALL HCHAR(21,14,130)
570 CALL HCHAR(21,15,128)
580 CALL HCHAR(21,16,128)
590 CALL HCHAR(21,17,131)
600 CALL HCHAR(22,14,132)
610 CALL HCHAR(22,15,133)
620 CALL HCHAR(22,16,128)
630 CALL HCHAR(22,17,134)
640 CALL HCHAR(23,15,135)
650 CALL HCHAR(23,16,136)
660 GOTO 660
665 REM Use FCTN 4 to stop
```

```
095 REM PROGRAM TWO
100 CALL CLEAR
110 CALL SCREEN(3)
115 REM Define characters patterns
116 REM for graphics
120 FOR X=128 TO 143
130 READ X$
140 CALL CHAR(X,X$)
150 NEXT X
155 REM Print LOGO info
160 PRINT TAB(10);"SUN CITY"::
170 PRINT TAB(10);"TI 99/4A"::
180 PRINT TAB(8);"COMPUTER CLUB"::::::::
::
185 REM Display graphics on screen
186 REM State of TEXAS
190 FOR Y=1 TO 35
200 READ R,C,X
210 CALL HCHAR(R,C,X)
220 NEXT Y
230 GOTO 230
235 REM Use FCTN 4 to stop
240 DATA FFFFFFFFFFFFFFFF,80808080808080
E0,000000FFFFFFFF3F,FCFCFCFCFEFFFFFF,1F1
F0F0703000000,FFFFFFBF3F3F3F1F,FFFFFFFEF
8F0C000
250 DATA 0F07070301000000,FEFCFCF8F8FC3C
18,00000000FF9F9FFF,00000000FFFFFFFF,CBD
BCBDBC9FF0F0F,1155117575FFFFFF
260 DATA 117515D511FFFFFF,04020100000000
00,0000000040201008,11,5,137,11,6,138,11
,7,138,12,5,139,12,6,140,12,7,141
270 DATA 13,5,142,13,6,143,14,6,142,14,7
,143,15,7,142,15,8,143,16,8,142,16,9,143
,17,9,142,17,10,143
280 DATA 18,10,142,18,11,143,19,11,142,1
9,12,143,20,12,142,20,13,143,21,13,142,2
0,15,128,20,16,129,21
290 DATA 14,130,21,15,128,21,16,128,21,1
7,131,22,14,132,22,15,133,22,16,128,22,1
7,134,23,15,135,23,16,136
```

TIPS FROM THE TIGERCUB

#40

Copyright 1986

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

Distributed by Tigercub Software to TI-99/4A Users Groups for promotional purposes and in exchange for their newsletters. May be reprinted by non-profit users groups, with credit to Tigercub Software.

Over 130 original programs in Basic and Extended Basic, available on cassette or disk, only $3.00 each plus $1.50 per order for PPM. Entertainment, education, programmer's utilities. Descriptive catalog $1.00, deductable from your first order.

Tips from The Tigercub, a full disk containing the complete contents of this newsletter Nos. 1 through 14, 50 original programs and files, just $15 postpaid.

Tips from the Tigercub Vol. 2, another diskfull, complete contents of Nos. 15 through 24, over 60 files and programs, also just $15 postpaid.

```
****************************
*                          *
*  Tips from the Tigercub  *
*  Vol. 3 is now ready.    *
*  Another 62 programs,    *
*  routines, tips, tricks. *
*  from Nos. 25 thru 32.   *
*  Also $15 postpaid. Any  *
*  two Tips disks $27 or   *
*  all 3 for $35 postpaid. *
*                          *
****************************
```

Nuts & Bolts (No. 1), a full disk of 100 Extended Basic utility subprograms in merge format, ready to merge into your own programs. Plus the Tigercub Menuloader, a tutorial on using subprograms, and 5 pages of documentation with an example of the use of each subprogram. All for just $19.95 postpaid.

Nuts & Bolts No. 2, another full disk of 108 utility subprograms in merge format, all new and fully compatible with the last, and with 10 pages of documentation and examples. Also $19.95 postpaid, or both Nuts Bolts disks for $37 postpaid.

Tigercub Full Disk Collections, just $12 postpaid! Each of these contains either 5 or 6 of my regular $3 catalog programs, and the remaining disk space has been filled with some of the best public domain programs of the same category. I am NOT selling public domain programs - my own programs on these disks are greatly discounted from their usual price, and the public domain is a FREE bonus!

TIGERCUB'S BEST, PROGRAMMING TUTOR, PROGRAMMER'S UTILITIES, BRAIN GAMES, BRAIN TEASERS, BRAIN BUSTERS!, MANEUVERING GAMES, ACTION REFLEX AND CONCENTRATION, TWO-PLAYER GAMES, KID'S GAMES, MORE GAMES, WORD GAMES, ELEMENTARY MATH, MIDDLE/HIGH SCHOOL MATH, VOCABULARY AND READING, MUSICAL EDUCATION, KALEIDOSCOPES AND DISPLAYS

For descriptions, send a dollar for my catalog!

The READFILE subprogram on my Nuts & Bolts #2 disk has a backward parentheses in line 21161. This is the corrected line -

```
21161 DISPLAY AT(17,1):"OPEN
 PRINTER #":"NAME? " :: ACCE
PT AT(17,15)VALIDATE(DIGIT)S
IZE(-3):P :: ACCEPT AT(18,7)
:P$ :: OPEN #P:P$ :: GOTO 21
163
```

When Texas Instruments developed Extended Basic, they took away the ability of Basic to redefine or color the characters in sets 15 and 16, ASCII 144 to 159, in order to make room in memory for sprites (they did let us have color set 0 instead. That is why Basic programs which use sets 15 and 16 will crash if you try to run them in XBasic.

Finally, John Behnke published in the Chicago Times newsletter an amazing routine which gave us back those missing sets. His routine was 13 sectors long. Recently, Richard Heath published in the L.A. newsletter a shortened version. And, without having any idea how it works, I have managed to scrunch it down to only 4 sectors -

```
1 CALL BXB
29999 !BXB by Jim Peterson,
adapted from VDPUTIL2 by Joh
n Behnke/Richard Heath
30000 SUB BXB :: CALL INIT :
: CALL LOAD(8194,37,194,63,2
40)
30001 CALL LOAD(16368,80,79,
67,72,65,82,37,58,80,79,75,6
9,86,32,37,168)
30002 !
30003 FOR J=1 TO 136 :: CALL
 LOAD(9529+J,ASC(SEG$(][\[]$
,J,1))):: NEXT J :: SUBEND
30004 SUB CHAR(A,A$):: CALL
LOAD(9500,A):: CALL LINK("PO
CHAR",A$):: SUBEND
30005 SUB COLOR(A,B,C):: CAL
L LOAD(9492,8,15+A,(B-1)*16+
C-1)
30006 CALL LINK("POKEV"):: S
UBEND
```

Note than line 30002 is missing. That's because there is no way to key it in. Once again we need a program that writes a program -

```
100 FOR J=1 TO 136 :: READ A
 :: M$=M$&CHR$(A):: NEXT J
110 OPEN #1:"DSK1.BXBDATA",V
ARIABLE 163,OUTPUT :: PRINT
#1:CHR$(125)&CHR$(0)&"][\[]$
"&CHR$(190)&CHR$(199)&CHR$(1
36)&M$&CHR$(0)
120 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1
130 DATA 2,224,37,20,3,0,0,0
,2,5,48,48,2,6,37,2,205,133,
2,134,37,17
140 DATA 17,252,4,192,2,1,0,
1,2,2,37,1,2,3,18,0,212,131,
4,32,32,20
150 DATA 208,4,9,80,2,32,3,0
,2,1,37,2,2,2,0,8,2,7,11,0,2
,8,7,0,193
160 DATA 1,192,193,193,180,9
7,133,145,135,21,1,113,136,6
,198,145
170 DATA 135,21,1,113,136,21
0,70,10,198,177,137,220,198,
2,131,37,10
180 DATA 17,240,4,32,32,36,1
6,6,2,224,37,20,3,0,0,0,4,32
,32,32,4
190 DATA 192,216,0,131,124,2
,224,131,224,4,96,0,112
```

RUN that to create a file BXBDATA on the disk. Then load the BXB program, and enter MERGE DSK1.BXBDATA. The unprintable line will pop into place. SAVE this completed BXB routine in MERGE format, and merge it into any Basic-only program. If you want, the result can be run ~~through a Compactor~~ program and turned into multi-statement program lines for more speed.

Or, you can write an Extended Basic program using all 16 character sets for graphics and color - actually 17, because set 0 is also available. Even the characters 24 through 31 can be redefined! Craig Miller has warned against fooling around in that area of memory, but there seems to be no problem with redefining the cursor (30) or the edge character (31).

Sprites can only use characters between 32 and 143 and their color cannot be changed with CALL COLOR(#_,_). I have not found any other bugs, but have not had time for much experimenting.

Here's an easy Tigercub challenge - run this one in Basic, not Extended Basic.

```
>LIST
1ØØ DISPLAY AT(1.1):Ø
>RUN
  Ø
  Ø
```

Why did it print the zero twice?

I wrote this next one primarily for blind users. It converts each PRINT or DISPLAY directly to speech output and also provides a speech prompt for INPUTs.

```
1ØØ !PRINT SPEAKER by Jim Pe
terson - to add OPEN #1:"SPE
ECH",OUTPUT and convert PRIN
T and DISPLAY statements to
PRINT #1
11Ø !Also writes a PRINT #1
for INPUT prompts
12Ø !Program to be converted
 must first be SAVEd in MERG
E format. Recommend it be RE
Sequenced before SAVEing, to
 make room for INPUT lines
13Ø PS$=CHR$(156)&CHR$(253)&
CHR$(2ØØ)&CHR$(1)&"1"&CHR$(1
81)
14Ø DISPLAY AT(3,1)ERASE ALL
:"INPUT FILENAME?":"DSK" ::
ACCEPT AT(4,4):IF$ :: OPEN #
1:"DSK"&IF$,INPUT ,VARIABLE
163
15Ø DISPLAY AT(5,1):"OUTPUT
FILENAME?":"DSK" :: ACCEPT A
T(6,4):OF$ :: OPEN #2:"DSK"&
OF$,OUTPUT,VARIABLE 163
16Ø PRINT #2:CHR$(Ø)&CHR$(1)
&CHR$(159)&CHR$(253)&CHR$(2Ø
Ø)&CHR$(1)&"1"&CHR$(181)&CHR
$(199)&CHR$(6)&"SPEECH"&CHR$
(179)&CHR$(247)&CHR$(Ø)
17Ø LINPUT #1:M$ :: P=POS(M$
,CHR$(156),3):: A=POS(M$,CHR
$(162),3):: Z=POS(M$,CHR$(18
1),3)
18Ø I=POS(M$,CHR$(146),1)::
IF I=Ø THEN 21Ø :: IF Z=Ø OR
 Z<I THEN PRINT #2:M$ :: GOT
O 24Ø
19Ø M2$=SEG$(M$,1,1)&SEG$(M$
,2,1)&PS$&SEG$(M$,I+1,Z-I-1)
&CHR$(Ø):: PRINT #2:M2$
2ØØ PRINT #2:SEG$(M$,1,1)&CH
R$(ASC(SEG$(M$,2,1))+1)&SEG$
(M$,3,255):: GOTO 24Ø
21Ø IF P+A=Ø THEN PRINT #2:M
$ :: GOTO 24Ø
22Ø M=MAX(P,A)
23Ø M$=SEG$(M$,1,2)&PS$&SEG$
(M$,M+1,255):: PRINT #2:M$
24Ø IF EOF(1)<>1 THEN 17Ø EL
SE CLOSE #1 :: CLOSE #2
25Ø DISPLAY AT(12,1)ERASE AL
L:"Type NEW and Enter" :: DI
SPLAY AT(15,1):"Type MERGE D
SK";OF$ :: END
****************************
```

MOLLY DARLING

```
1ØØ CALL CLEAR :: CALL SCREE
N(5):: FOR SE=1 TO 12 :: CAL
L COLOR(SE,16,5):: NEXT SE
11Ø DISPLAY AT(3,8):"MOLLY D
ARLING": :" Written and perf
ormed by": :TAB(9);"Eddy Arn
old" :: DISPLAY AT(24,1):"Pr
ogrammed by Jim Peterson"
12Ø FOR D=1 TO 2ØØ :: NEXT D
 :: DISPLAY AT(12,1):"Just a
 moment.......": :".....look
ing for my music..."
13Ø DIM N(1ØØ),N2(1ØØ),A(25Ø
),B(25Ø),C(25Ø):: F=11Ø :: F
OR J=1 TO 8Ø :: N(J)=INT(F*1
.Ø59463Ø94^(J-1)+.5):: NEXT
J
14Ø DATA 16,11,8,16,8,11,16,
4,11,18,11,8
15Ø DATA 2Ø,16,11,23,11,16,2
5,21,16,28,16,21
16Ø DATA 23,2Ø,16,23,16,2Ø,2
3,11,16,23,16,11
17Ø DATA 2Ø,11,16,2Ø,16,11,2
Ø,8,11,2Ø,11,8
18Ø DATA 2Ø,11,16,25,16,11,2
3,11,16,2Ø,8,4
19Ø DATA 18,16,1Ø,18,1Ø,16,1
8,16,1Ø,18,11,16
2ØØ DATA 18,15,11,18,9,15,18
,11,9,18,9,3
21Ø DATA 28,8,1,28,13,8,28,8
,13,28,13,4
22Ø DATA 27,2Ø,18,27,18,2Ø,2
Ø,18,12,2Ø,12,18
23Ø DATA 25,21,16,25,16,21,2
5,13,16,25,16,13
24Ø DATA 27,23,21,27,21,23,2
7,23,18,27,18,21
25Ø DATA 28,23,2Ø,28,2Ø,23,2
8,2Ø,16,27,16,2Ø
26Ø DATA 3Ø,21,13,28,13,21,2
7,21,13,25,13,21
27Ø DATA 23,2Ø,16,23,16,2Ø,2
Ø,11,16,2Ø,16,11
28Ø DATA 3Ø,23,13,28,13,23,2
3,2Ø,13,2Ø,13,16
29Ø DATA 25,21,16,25,16,21,2
5,21,16,27,16,21
3ØØ DATA 28,23,2Ø,2Ø,16,11,1
8,15,11,2Ø,11,15
31Ø DATA 16,11,8,16,8,11,16,
9,1,16,1,9
32Ø DATA 16,11,8,16,8,11,16,
1,8,16,13,1
33Ø DATA 25,21,16,25,16,13,2
5,13,9,25,9,4
34Ø DATA 23,2Ø,16,23,16,11,2
3,11,8,23,8,4
35Ø DATA 21,18,11,21,11,9,21
,9,6,2Ø,6,3
36Ø DATA 21,16,11,2Ø,16,11,2
Ø,11,8,2Ø,8,4
37Ø DATA 18,13,1Ø,18,1Ø,6,18
,6,1,2Ø,13,1Ø
38Ø DATA 22,18,13,28,22,18,2
7,18,22,25,22,18
39Ø DATA 23,18,15,23,15,11,2
3,11,6,23,6,3
4ØØ DATA 23,21,15,23,15,11,2
3,11,9,23,9,6
41Ø DATA 16,13,8,16,8,13,16,
13,8,18,13,9
42Ø DATA 2Ø,11,8,21,8,11,2Ø,
11,8,18,11,6
43Ø RESTORE 14Ø :: T=16 :: G
OSUB 48Ø :: RESTORE 14Ø :: T
=4 :: GOSUB 48Ø :: RESTORE 1
8Ø :: T=12 :: GOSUB 48Ø :: R
ESTORE 14Ø :: T=16 :: GOSUB
48Ø
44Ø RESTORE 21Ø :: T=28 :: G
OSUB 48Ø :: RESTORE 17Ø :: T
=4 :: GOSUB 48Ø :: RESTORE 2
5Ø :: T=4 :: GOSUB 48Ø :: RE
STORE 28Ø :: T=4 :: GOSUB 48
Ø :: RESTORE 19Ø :: T=8
45Ø GOSUB 48Ø :: RESTORE 14Ø
 :: T=16 :: GOSUB 48Ø :: RES
TORE 29Ø :: T=48 :: GOSUB 48
Ø :: RESTORE 14Ø :: T=16 ::
GOSUB 48Ø :: RESTORE 41Ø ::
T=8 :: GOSUB 48Ø
46Ø RESTORE 31Ø :: T=8 :: GO
SUB 48Ø :: GOTO 49Ø
47Ø GOTO 49Ø
48Ø FOR J=1 TO T :: X=X+1 ::
 READ A(X),B(X),C(X):: A(X)=
A(X)+12 :: B(X)=B(X)+12 :: C
(X)=C(X)+12 :: NEXT J :: RET
URN
49Ø DISPLAY AT(1Ø,1):"Contro
l volume of 3 voices":"using
 1, 2 and 3 keys for":"loude
r and Q, W and E for":"softe
r.":""
5ØØ DISPLAY AT(15,1):"Contro
l speed using 'F' for":"fast
er and 'S' for slower."
51Ø DISPLAY AT(18,1):"Change
 key using 'A' for":"higher
and 'D' for lower."
52Ø DISPLAY AT(21,1):"Press
'Z' for minor key, 'X'":"for
 major key." :: V1,V2,V3=1Ø
:: F,P,Y=Ø :: X=2ØØ
53Ø FOR J=1 TO 192 :: CALL S
OUND(-999,N(A(J)-Y),V1,N(B(J
)-Y),V2,N(C(J)-Y),V3):: FOR
T=1 TO X/5Ø :: P=1^X :: NEXT
 T
54Ø CALL KEY(Ø,K,S):: IF S<1
 THEN 71Ø :: ON POS("123QWEF
SADZX",CHR$(K),1)+1 GOTO 71Ø
,55Ø,56Ø,57Ø,58Ø,59Ø,6ØØ,61Ø
,62Ø,63Ø,65Ø,67Ø,69Ø
55Ø V1=V1-1-(V1=Ø):: GOTO 71
Ø
56Ø V2=V2-2-(V2=Ø)*2 :: GOTO
 71Ø
57Ø V3=V3-2-(V3=Ø)*2 :: GOTO
 71Ø
58Ø V1=V1+2+(V1=3Ø)*2 :: GOT
O 71Ø
59Ø V2=V2+2+(V2=3Ø)*2 :: GOT
O 71Ø
6ØØ V3=V3+2+(V3=3Ø)*2 :: GOT
O 71Ø
61Ø X=X-2Ø-(X<2)*2Ø :: GOTO
71Ø
62Ø X=X+2Ø :: GOTO 71Ø
63Ø IF F=1 THEN GOSUB 7ØØ
64Ø Y=Y-1-(Y=-2Ø):: GOTO 71Ø
65Ø IF F=1 THEN GOSUB 7ØØ
66Ø Y=Y+1+(Y=6):: GOTO 71Ø
67Ø IF F=1 THEN 71Ø :: GOSUB
 68Ø :: GOTO 71Ø
68Ø F=1 :: Y=Ø :: FOR W=3 TO
 27 STEP 12 :: N2(W)=N(W)::
N(W)=N(W-1):: N2(W+5)=N(W+5)
:: N(W+5)=N(W+4):: N2(W+1Ø)=
N(W+1Ø):: N(W+1Ø)=N(W+9):: N
EXT W :: RETURN
69Ø IF F=Ø THEN 71Ø :: GOSUB
 7ØØ :: GOTO 71Ø
7ØØ F=Ø :: FOR W=3 TO 27 STE
P 12 :: N(W)=N2(W):: N(W+5)=
N2(W+5):: N(W+1Ø)=N2(W+1Ø)::
 NEXT W :: RETURN
71Ø NEXT J :: J=192 :: FOR V
=1Ø TO 3Ø :: CALL SOUND(-999
,N(A(J)-Y),V,N(B(J)-Y),V,N(C
(J)-Y),V):: NEXT V :: FOR D=
1 TO 5ØØ :: NEXT D :: GOTO 5
3Ø
```

MEMORY FULL

Jim Peterson

TIPS and TECHNIQUES

This month's Tips and Techniques comes to us by way of Nagy Zoltan. He has scraped up a bunch more pokes and peeks to add to the list we were given last month by Dale.

CALL LOAD(-31745,0)	Produces a frozen screen, and then blanks the screen. Restore with FCTN(-)
CALL LOAD(-31748,0 to 255)	Changes the cursor flashing and response tone rate.
CALL LOAD(-31794,255 to 0)	Timer for CALL SOUND (counts from 255 to 0)
CALL LOAD(-31873,3 to 30)	Screen column to start at with a PRINT statement.
CALL LOAD(-31884,0 to 5)	Keyboard mode for CALL KEY statement
CALL LOAD(-32572,128)	Disables keyboard
CALL LOAD(-32729,0)	Run DSK1.LOAD

And there you have it, a few more to add to your list.

GRAM CARD REVIEW

I was fortunate to have been invited over to Heino's house to get a first hand look at the capabilities of the GRAM card. Here are some of my observations.

My first question, and the question others have raised is "Does the GRAM card speed up the functions of the console, especially in the Basic environment ?". I am sorry to report that it doesn't, since it is operating as GRAM using the GRAM/GROM bus which is inherently slower than the CPU bus. A look at the schematics will show you why. This is a very minor shortcoming considering the capabilities of the card.

The most useful function of the card is the card's capability to convert your command modules to a disk file(s). This allows the cumbersome modules to be stored away out of the way, while their programs reside on a few disks. It does of course take a little longer to get the programs running, but is well worth it. The firmware on the card has the support to allow you to off load the modules to disk. The only extra effort required is a small loader program, which can be done from a Basic program. The loader is a one time deal, once it is created all it takes is one disk with the loader programs and the module programs to get the GRAM card loaded with your desires. I was really surprised with the ease at which a module can be converted. The manual is very concise in this conversion process, and takes you by the hand througn the whole process, they have even included the loader program example. It may be easier though to create the loader program from the Editor/Assembler Editor, since the loader is in Display Variable 80 format, and is only a few lines, but this is not discussed in the manual.

Now, since your modules are now loaded in to RAM there will be the temptation to modify the program to suit your needs. With a knowledge of the GPL language this is entirely possible. Mechatronic has even made this task easy by including a subroutine that allows you to modify GRAM memory contents to suit your taste. The folks at Mechatronic didn't stop there though, they also allow VDP and CPU memory modification from the subroutine. Whether you use it to modify GRAM memory or not, it is a nice subroutine to have available just for the capability of VDP memory modification.

There is also a function that could be used for product development. That is it's capability to convert your favorite program to GRAM format. Thus if you had built the Ryte Data Eprom programmer you could take your favorite basic program, convert it to GRAM format, burn an EPROM from the GRAM contents, and sell it to your friends. Of course you would have to package it inside a GROM emulator module. This is the only shortcoming of the manual, Heino an I tried to convert a program that wouldn't function after the conversion process. Heino found out later that the GRAM card has to be fully reset before the conversion is attempted. Heino converted a program after finding this out, and reports that the process works as designed. Of course not everyone will want to sell their programs, and that is not the real purpose of the conversion process. What it allows is for your program to loaded and appear as a selection on the master title screen, and can be run from there. This can only be done, though, from regular Basic. Extended Basic programs will not convert. You also lose the RESTORE statement, as a program with a RESTORE statement will not convert properly.

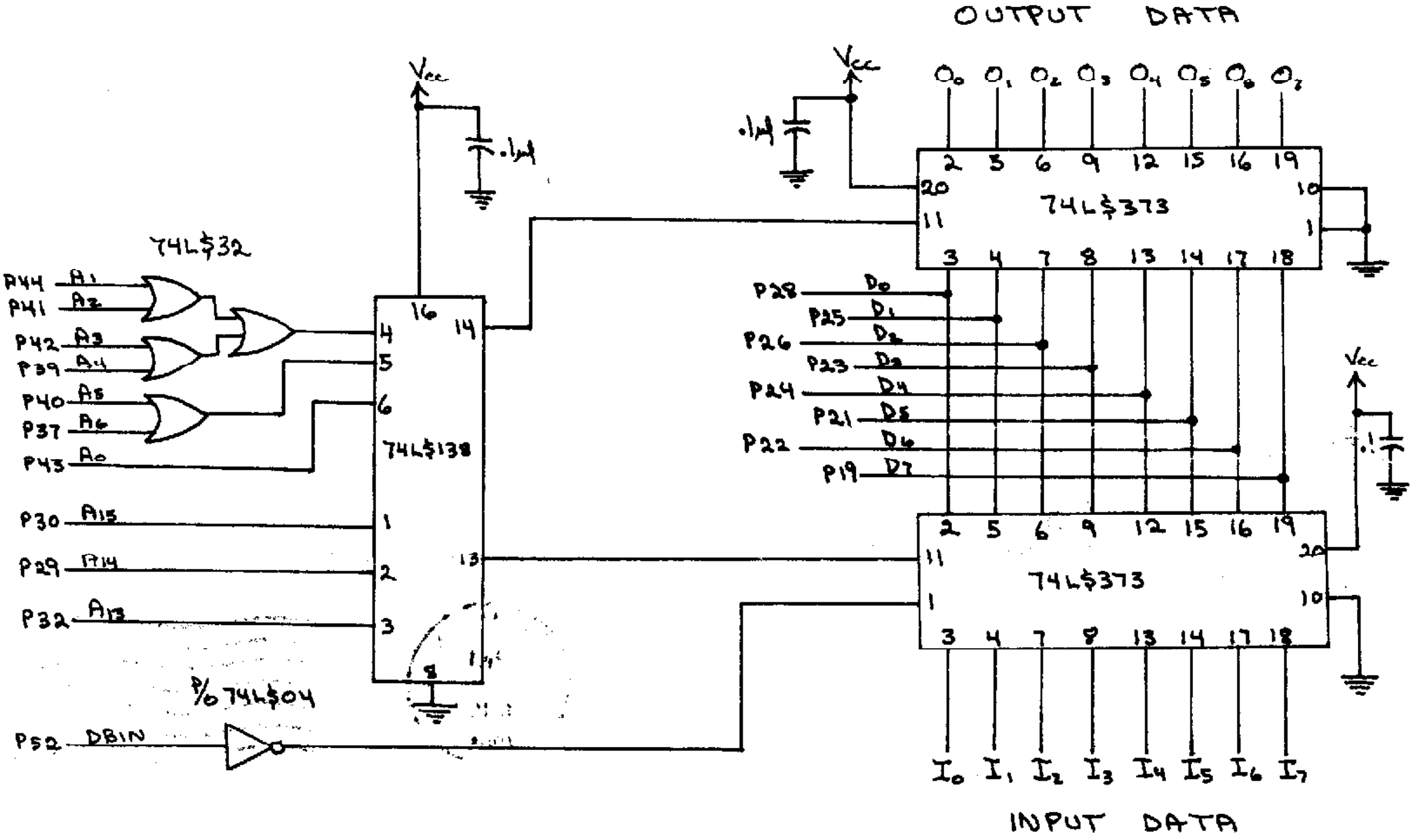
If this is not enough you also have the capability to load GPL programs that you assemled with a GPL assembler package, or you can load the RAM area with tagged object code that you have also developed. Both of these types of programs have to use absolute addressing, and all Utilities are up to you to write, as the normal utilities are not available.

The card itself is a well made product, and the Mechatronic engineers have put in a DIP switch so you have the capability of changing it's CRU base address, something a lot of developers don't do. The firmware built into the card supports up to eight GRAM cards, but Basic can only access up to three cards. The card Heino has is the 128K byte version, this is partioned as 64K of GRAM and 64K of RAM. I can't quote the exact price because frankly I don't know. If you need more info on the card contact Ryte Data in Canada, I beleive they are the only North America distributors for this product.

THE CARBON CONVERTER

I am sorry to report that due to parts availability the project I was going to present this month won't be up and running. So instead what I am going to do is present, for you true blue hardware enthusiasts, is a partial schematic that has not had any bugs worked out of it. It may not even function at all. So get out your data books and dig in. It was originally designed for use in the Expansion box, so it is up to you to add the RDBENA signal (Remote Data Bus ENable). While you have out your data books check for proper component selection. Maybe there are some bad choices for components, maybe not. It should be able to work within the bounds of the previously decoded address space (>8000 to >82FF).

Should my parts come in soon, I will have the project in next month's newsletter.



The Sun City TI-99/4A User's Group newsletter is published monthly. The editor for the newsletter is the club president all inquiries, comments, or complaints should be addressed to him/her at the club mailing address. The newsletter is distributed free of charge to other User's Groups in exchange for their newsletter. Individual subscriptions to the newsletter are given with membership to the Sun City User's Group.

Articles may be reprinted by other newsletters as long as credit is given to the Sun City Users Group and the article author(s).

Advertising rate is $5.00 for a full page.

The Sun City User's Group meets the first Monday of each month at the El Paso Trade School 4710 Alabama St. at 7 PM. Ask the person at the front desk of the Trade School for the room number of the meeting. All interested parties are encouraged to attend.

********************************* NEWSLETTER 17 ************************************

SUN CITY TI 99/4A COMPUTER CLUB
P.O. BOX 6966
EL PASO, TEXAS 79906

Dallas TI Home Comp. Group
1221 Rosewood Place
Irving, Texas 75061

THE FINE PRINT

THE VIEWS AND OPINIONS REFLECTED IN ARTICLES OF THIS NEWSLETTER ARE THOSE OF THE ARTICLE AUTHOR, AND DO NOT NECESSARILY REFLECT THE VIEWS OF THE SUN CITY TI 99/4A COMPUTER CLUB. IN NO EVENT WILL THE SUN CITY TI 99/4A COMPUTER CLUB OR THE AUTHORS OF ARTICLES BE HELD LIABLE FOR DAMAGES RESULTING FROM USE OF INFORMATION CONTAINED IN THIS NEWSLETTER.