CEDAR VALLEY 99er UG JULY 1992

PRESIDENT: Jack Johns (319) 366-4541
VICE PRES: Wayne Betts (319) 377-2493
TREASURER: Bruce Winter (319) 393-0610
SECRETARY: Jim Green (319) 377-4073
NL EDITOR: Gary Bishop (319) 377-9574
LIBRARIAN: Bob Heiderstadt (319) 927-4215

CEDAR RAPIDS/MARION

Supporting the TI-99/4A and Geneve 9640 in Eastern Iowa

NOTICE: MEETING IS MOVED TO WEDNESDAY

NEXT MEETING: 6:30 PM JULY 15, 1992

WEST MUSIC, COLLINS ROAD SQUARE

CONTENTS:

Cedar Valley Amateur Radio Club Ham Fest

August 16, 1992, Teamsters Hall.

5000 J St. SW, Cedar Rapids, IA

Admission: $4.00 Open 8 AM - 4PM.

Teamster's parking lot for tailgate/
trunk sales and handicapped parking only.

MINUTES OF THE JUNE MEETING

The June meeting of the Cedar Valley 99ers User Group was held on June 9 with a normal summer attendance of seven members. (Ed was missed; probably working the fields...) We began with an informal discussion about exchange newsletters from other user groups around the country. We agreed to discontinue mailing our newsletter to a couple of groups that we have not heard from in ages, and a suggestion was made to initiate an exchange with the Lima, Ohio group. (Yours truly has that action item, but it has not been acted upon yet.) A ham/computer swap meet was held in Cedar Rapids, but the weather kept the attendance down. About twenty sellers hung around until the rain showed up, so it was not an exciting event, according to our eye witnesses.

More to the point of TI business, we heard about a disk of 110 subograms compiled/written by Jerry Stearn, called SubIndex. This disk was advertised in MICROendium for only $6.00, and it proved to be worth the money. Included are subroutines for sound effects, printer dumps, statistics, menus, random number generators, text/string manipulators, record keeping, and more! See MICROpendium for the ordering address.

The club recently received a full disk of text from Jim Peterson, for use in our newsletter. Our editor finds that there is more than enough articles from Jim to fill several newsletters (thanks, Jim!), so he offered to make a copy of the disk for those who don't want to wait several issues to read all of Jim's work. See Gary for your copy.

An informal demo was given of disk # 396 from our library, which is a Japanese language tutor. We had fun trying to guess the katakana and hiragana characters for each syllable (one of us had a cheat sheet!). Check this disk out if you have an interest in foreign languages. We then had a more formal demo of Barry Traver's UNBASHER, which will separate XB multiple command programs into programs written on individual lines. This will make programs much easier to edit, and could allow the running of an XB program in console BASIC, assuming the last two character sets aren't called for in the program.

Moving to the formal business portion of the meeting, it was moved and seconded that the group dispense with the formal business meeting during the June-August meetings. Motion passed 6-1. No further business was conducted.

The evening ended with a tribute to Lawrence Welk. John Johnson played a resounding version of the Beer Barrel Polka that was downloaded from the Boston Computer Society BBS. Since John is the most active member on bulletin boards, he was asked to sign on to the C.O.N.N.I. Clearinghouse BBS to find out why our group has not been informed of the procedures for participating in that Clearinghouse of group newsletters. John will report progress at the July meeting.

Ahead of deadline,
Jim Green, secretary

TURBO SPEECH
(or How to Speed up the spoken word)
by Stephen Shaw
(Excerpted from the TI99/4A Exchange TI$MES of Great Britian, issue #6.84 via HOCUS 99 newsletter Dec 1991)

Now on to something really juicy. SPEECH. Old hat huh? Well, this information will give you speech in TI Basic with the Mini Memory, or if you have XBASIC with 32K RAM, will give you speech just a bit faster than using CALL SAY which slows programs down to no end.

For this information I am indebted to Neil Lawson who has been delving. Speech requires either:

XBASIC with 32K memory or Mini-Memory and the Speech Synthesizer.

Program framework (For timing purposes):

```
20 CALL INIT
30 S=27648
100 FOR I=1 TO 1000 :: NEXT I
110 PRINT "START....."
120 FOR X=1 TO 20
130 REM TEST ROUTINE HERE
140 FOR T=1 TO 30
150 PRINT ">":
160 NEXT T
170 NEXT X
180 PRINT "END....."
```

This standard routine sets up a framework to test our new routine in, and gives a basic time reference.

(NB: Times quoted are for my system: yours may be different, but the ratios should be similar.)

Running the above program, with the loops in line 140 running 30 times as shown, takes 18.7 seconds from "START" to "END". Change line 140 to loop just 20 times and the timing is 12.7 seconds.

Now we can insert our two possibilities:

The first is available only in XBASIC:

```
130 CALL SAY("#THAT IS INCODRECT#")
```

Run this program again: If line 140 is looped 20 times, the time is 44 seconds. If line 140 is looped 30 times, the time is 50 seconds.

The time for the speech is constant, it adds about 21 seconds to the program.

Now for something different, (also works with Mini-Memory):

```
130 CALL
LOAD(S,70,"",S,65"",S,72,"",S,70,"",S,64
,""S,80)
```

If, you now run the program, it says the same thing as many times, but look at the timing:
If line 140 loops 20 times: 26.3 seconds.
If line 140 loops 30 times: 26.5 seconds.

We know that looping line 140 an extra 10 times adds 6 seconds...so where have those 6 seconds gone?

The CALL SAY routine holds everything up until it has finished speaking. But using the CALL LOAD equivalent, while the computer is speaking, it gets on with the next chore too. The "dead time" is used and soaks up these 6 seconds.

Thus using the CALL LOAD equivalent, the computer speaks faster, and also permits your program to run more quickly if there is work for it to do between speech outputs.

That's the clever demonstration! (Impressed"?) Now for the theory.

References: Editor/Assembler Manual, pages 351, 355, 422-427. Reference in para 1, page 355, should be the Section 22.1.4 not as printed in the manual.

Address -27648 is the SPEECH WRITE address. We keep feeding it with bytes, and in due course the computer speaks. The bytes to feed to that address are found out as follows:

First, decide what you want to say from the standard vocabulary. Then look in the table (pp. 422-427) for the address of that word or phrase. "THAT IS CORRECT" is given as 6816. That is Hexadecimal not a Decimal number. The four numbers are reversed, and become 6168.

Now we offset them by Hex 40 and feed them in. As we are dealing with decimals with our CALL LOAD, that means we add decimal 64 to each digit in turn:

(6+64) (1+64) (6+64)
70 65 72 70

If the numbers were Hex A-F these have a decimal value as follows:

A=10 B=11 C=12 D=13 E=14 F=15

Now we must indicate end of word by loading a zero, again offset, thus 0+64=64. Finally, instruct the computer to speak by loading Hex 50, Decimal 80.

Thus we have loaded, in order:

Check back to the listing. Note the way CALL LOAD has been used: a simple command to load the same address with several different values.

To assist your experimentation, here are some Hex addresses from the manual. Remember to [reverse] them, translate to decimal and offset.

```
TEXAS INSTRUMENTS...6696 THAT IS
RIGHT....68FE
WHAT WAS THAT.......77E9 READY TO
START....56B3
YOU WIN.............7DDB
AGAIN....17A5..
ANSWER..............1913 CHECK...1D82
CHOICE..............1DA2 COMMAND.1F1A
ELSE................28B6 GOODBYE.3148
HELP................3571 HURRY...3757
```

END

The following items were discovered by Jim Peterson, and relayed to me in correspondence. Interesting info on exploring extended Basic:

```
90 CALL CLEAR :: PRINT TAB(7
);"SPRITE PUZZLE #1":"
  from Tigercub"
100 PRINT "A non-existent sp
rite can be":"created by CAL
L MOTION.": :"It apparently
starts in"
110 PRINT "dot-row 1, dot-co
lumn 1, and":"has color 1, b
ut its pattern":"is not that
 of any ASCII!"
120 !by Jim Peterson
130 FOR CH=0 TO 255 :: PRINT
 CHR$(CH);:: NEXT CH
135 PRINT "CALL MOTION(#1,5,
5):: CALL COLOR(#1,16):: CAL
L MAGNIFY(4)"
140 CALL MOTION(#1,5,5):: CA
LL COLOR(#1,16):: CALL MAGNI
FY(4)
150 GOTO 150
```

And another -

```
100 DISPLAY AT(3,5)ERASE ALL
:"SPRITE PUZZLE #2": :"
 from Tigercub"
110 DISPLAY AT(7,1):"Non-exi
stent sprites can be":"creat
ed by CALL COLOR.": :"Their
existence can be con-"
120 DISPLAY AT(11,1):"firmed
 by CALL COINC, but":"CALL P
OSITION reports that":"they
have no position!"
130 CALL COLOR(#1,16):: CALL
 COLOR(#2,16)
140 CALL COINC(#1,#2,1,X)::
DISPLAY AT(15,1):"COINC #1,#
2=";X :: CALL POSITION(#1,X,
Y)
150 CALL POSITION(#1,X,Y)::
DISPLAY AT(17,1):"POSITION #
1=";X;Y
160 CALL POSITION(#2,X,Y)::
DISPLAY AT(19,1):"POSITION #
2=";X;Y
170 IF FLAG=1 THEN 140 :: FL
AG=1
180 DISPLAY AT(21,1):"PRESS
ANY KEY"
190 CALL KEY(0,K,S):: IF S=0
 THEN DISPLAY AT(21,1):"pres
s any key" :: GOTO 180
200 DISPLAY AT(21,1):"Until
they're set in motion!"
210 CALL MOTION(#1,5,5):: CA
LL MOTION(#2,-5,-5):: GOTO 1
50
```

I recently programmed a diskfull of gospel songs, and in each one I used this formula to set up an array containing the frequencies for 3 octaves:

```
DIM N(36) :: F=110 :: FOR J
=1 TO 36 :: N(J)=INT(F*1.059
463094^(J-1)+.5):: NEXT J
```

At the end of each selection I put CALL INIT :: CALL LOAD(-31961,149) I don't remember where I learned that one, but it clears the screen, sets all colors and characters to default, deletes sprites, and looks for a LOAD program on DSK1.

The LOAD program has a routine to play each song one after another, but one song crashed with a BAD VALUE error even though it had previously been OK. I found that this was the only song that actually used N(1). The value should have been 110 but it had somehow changed to 24263 which the program line multiplied by 2, therefore out of range.

I found that the routine was correctly giving N(1) a value of 110 the first time but after the CALL LOAD it always had the 24263 value. Substituting other values for 110, I found that any value was being multiplied by 220.57000101 rounded off.

Further experimentation revealed that the problem was being caused by the ^ (exponentiation sign, shift 6 on your keyboard, in case someone prints this through the Formatter!). So I wrote this little routine to experiment with:

```
100 FOR J=1 TO 10 :: PRINT
2^J  :: NEXT J :: CALL INIT
 :: CALL LOAD(-31961,149)
```

I saved that as DSK1.TEST and then wrote another one 100 RUN "DSK1.TEST", saved that as DSK1.LOAD, and then entered RUN "DSK1.TEST".

It printed out the proper values time after time, so I changed the 2^J to read 2^(J-1). The first time around, the first value was 1 as it should be - the computer will consider any number to the power of 0 to have a value of 1. But, the next time around, the first value was F0.57000101 !

That was not even a valid numerical representation, so I changed the formula to 2^(J-1)*2, expecting it to crash. Instead, it gave me a value of 441.140002 !

Further experimentation showed that 2^(J-1)+1 gave a value shown as 1<1.570001.

Changing the +1 to +10 gave 1=0.570001 and to +100 gave 2<0.570001 !

So, poking a value of 149 into -31961 will cause any number taken to the power of zero to have a value of 220.57000101 which will be represented on screen in some apparently undocumented format - it's not even radix 100. I wonder if the fellows who built this computer could explain that!

In Tips #62 I reported on the weird behavior of the CALL LOAD(-31961,149), when used to clear all defaults and search for a LOAD file on DSK1. I have since found that if you put this CALL at the beginning of a program, it will not execute until an END or STOP is reached - but if you break the program with FCTN 4, it will not be in memory!

I stated that after this CALL LOAD was executed, any number taken to the power of 0 (which should be a value of 1) acquired a value of 220.5727273. I was led astray by the INT in the the formula in which I first found this puzzle. Actually it is 220.57000101, which prints to the screen in the peculiar format F0.57000101.

If a number between 1 and 9 is added to that, it is printed as 1< followed by the number being added, followed by the decimal part. For a number between 10 and 19, the < is changed to = and between 20 and 29 it becomes > (note the ASCII sequence); from 30 to 35 it becomes ? but from 36 to 99 99 the decimal portion is preceded by 0 to 63 respectively. 100 is 2<0.570001 and the pattern continues.

Although these are not valid representations of numbers, they are treated as such. Run a program to give N the power of 2^0, then break the program and experiment in immediate mode.

PRINT N gives that strange F0.57000101. PRINT N+1, or whatever, gives values represented in the format described above. PRINT N*1 will give the true numeric value 220.57000101 but multiplying by some other values gave me results in the odd format, as did dividing.

Peter Walker pointed out to me that trying to subtract from N within a program resulted in printing a value followed by a crash reporting a SYNTAX ERROR (in the line which had just been executed!) followed by a jump to a non-existent line zero!

NEXT REGULAR MEETING: Wednesday

July 15, 1992 6:30 PM

WEST MUSIC COMPANY
COLLINS RD. SQUARE, MARION
NORTH OF LINDALE MALL

> > > Note changed day < < <

N-1 should be 219.57.. of course, but in immediate mode PRINT N-1 results in 63.57000101. In the format in which added values are printed, this would be 319.57000101 but the 63.. is actually a decimal value, as can be proved by PRINT CHR$(INT(N-1))! When I tried to get a zero value by PRINT N-64.57000101, the computer blew its mind.

Does anyone know what is going on here?

Cedar Valley 99'er Users Group
c/o Jim Green
377 Cambridge Dr. NE
Cedar Rapids, Iowa 52402-1446

FIRST CLASS

Send To:

GARY BISHOP
124-222
3270 28TH AVE
MARION, IA 52302