Winnipeg June 86

Newsletter

June's Newsletter

The Winnipeg 99/4 User Group is a non-profit organization formed by computer hobbyists for users of the Texas Instruments 99/4A Home Computer and compatables. The content of this publication does not necessarily represent the view of the Winnipeg 99/4 User Group.

Next General Meeting - Date : September 4th, 1986
Time : 7:00 P.M.
Place: Winnipeg Centennial Library
1st Floor, Assembly Room

Executive 1986:

President:	Jim Bainard	334-5987
Treasurer:	Bill Quinn	837-7758
Newsletter Editor, Book and User Programs Librarian:	Mike Swiridenko	772-8565
Contributing Editor:	Paul Degner	586-6889
Inter-Group Representative and Newsletter Publisher:	Dave Wood	895-7067
Asst. Newsletter Publisher:	Hank Derkson	
Systems Co-Ordinator:	Rick Lumsden	253-0794
Educational Co-Ordinator:	Sheldon Itscovich	633-0835
Public Domain Librarian: 822 Henderson Hwy.	Gordon Richards	668-4804
Module Librarian:	Peter Gould	889-5505

Mailing Address: NEWSLETTER EDITER
WINNIPEG 99/4 USERS GROUP
P.O.B. 1715
WINNIPEG, MANITOBA
CANADA, R3C 2Z6

EDITORIAL COMMENTS:

Well summer is just around the corner! (If you don't believe me go to the nearest corner and have a look.) This means that our club's meetings will break for the season and return again in September. The executive wishes to inform the club's membership that the fall meetings, due to declining membership, will be held in the smaller lower level meeting room of the Wpg. Public Library. The executive would also like to ask the membership if they would like to continue the formal meetings or if the our club should formally disband. Discussion of the future of our club will be held during this month's meeting. One suggestion raised at previous executive meetings was that the club should continue until the end of the year.

To make for light summer reading this month's newsletter will be rather brief. Check out Hints and Tips for TI-WRITER transliteration commands. Helpfile will discuss the XBasic random statement, Forth DO loops, and Assembler Jump insttructions. Please read Miscellania for more club information. Have a decent summer! See you all again on the fall!

If you have a review, user hints, or helpful programming tips, get them to me for the next newsletter. The deadline that I have set for submissions is one week before the date of the group's meeting. Thanks go to all who have submitted items for this issue of our newsletter.

MISCELLANIA:

Miscellaneous news and reminders.

1.) As discussed in my comments, the fall meeting will be held in the lower level meeting room of the Library.

2.) The member written library is to be terminated. Lack of input into this library have contributed to this decision. Most of the programs in this library will be turned over to the public domain library, pending release by authors.

3.) Thanks goes to Mark Gibson of the MAD HUG users group for passing along PILOT and documentation to our group. We look forward to further trade. PILOT is another FREEWARE language originally designed for Computer Assisted Instruction (CAI).

4.) FREEWARE... I have recently completed a CRIBBAGE program and wish to have it distributed as FREEWARE. The program is in Extended Basic and requires the 32K mem-exp. It features on/off of counting messages, as well as a cheat function. If you are interested Mail the newsletter editor, Mike Swiridenko, at the address on the cover of this newsletter. Send a single diskette, mailer, and appropriate return postage.

5.) Welcome to our mailing list:
the METRO-JACKSON MICRO-COMPUTER USERS GROUP
the GREATER ORLANDO 99'ERS USER'S GROUP

HELPFUL HINTS AND TIPS!
(FOR THE USERS, BY THE USERS!)

This column features tips brought to my attention from members of this group, other user group's newsletters, and various other sources. **WARNING:** These hints and tips are to be used at your own risk!

FAST-TERM:

To use the XModem file transfer of this program, press FCTN CTRL N to specify a filename for the UP/DOWN load operation. After having specified the file, press FCTN CTRL X to activate the XModem file transfer routine. Follow the instructions and you should have no problems.

XModem is a file transfer protocol that was developed for the transfer of data between microcomputers. XModem provides error checking, of transmitted data, by means of a CRC (Cyclic Redundancy Code) value. Data is transmitted in fixed size blocks, each with their own CRC value. The CRC value is basically the sum of the bits in the transmitted data.

To check for transmission errors, the bits of the transfered data are summed and compared to the CRC value sent with the data. If the sum of the bits in the received data does not match the transmitted CRC value then data is assumed to have been lost during transmission, and a retransmission is requested. If the sum and CRC values do not match after several retries the transfer is aborted. Faulty transfers can be caused by noisy phone lines, weak modem signals, or other hardware errors.

TI-WRITER

The following transliteration commands were taken from an article by David Reed in Volume NO.60173 NO.2 of the 99/4a National Assistance Group newsletter.

Using the transliterate command, you can set certain characters on the keyboard so that whenever these characters are encountered the printer will be set in a certain typestyle. The following are examples of some transliterate commands you could use.

NAME	COMMAND ON	COMMAND OFF
EMPHASIZED	.TL 123:27,69	.TL 125:27,70
DOUBLE STRIKE	.TL 91:27,71	.TL 93:27,72
SUPERSCRIPT	.TL 62:27,83,0	.TL 94:27,84
SUBSCRIPT	.TL 60:27,83,1	.TL 94:27,84
ITALICS	.TL 33:27,52	.TL 63:27,53
EMPHASIZED ON AND DOUBLE STRIKE	.TL 123:27,69 .TL 91:27,71	.TL 125:27,70 .TL 93:27,72

EMPHASIED ON AND ITALICS	.TL 33:27,52 .TL 123:27,69	.TL 63:27,53 .TL 125:27,70
DOUBLE WIDE	.TL 35:27,87,1	.TL 37:2787,0
COMPRESSED TYPE	CTRL U SHIFT O	CNTL U SHIFT R

****

Using the transliterate commands above, if you want to print in EMPHASIZED, you would type a "{" before the text you want printed is emphasized, then format the document. When it is printed, all of the text following the "{" will be emphasized. The "{" will not be printed. These typestyles stay set until they are turned off or changed by another command, or the printer is turned off.

Some of these typestyles can be combined and others cannot. Check your printer manual for those which can be combined.

ASSEMBLY LANGUAGE

The following is a simple subroutine that will give a BAD RESPONSE TONE when called. To call this routine use the statement- BL @HONK

```
HONK  CLR   R0
      MOVB  R0,@>837C        CLEAR THE STATUS BYTE.
      BLWP  @GPLLNK          BRANCH AND LINK TO THE GPL LIBRARY.
      DATA  >0036            OFFSET TO BAD RESPONSE TONE S/R.
      CLR   R0
HK1   MOVB  @>83CE,R0        LOOP TO SEE IF SOUND HAS FINISHED.
      LIMI  2                ENABLE/DISABLE VDP INTERRUPT.
      LIMI  0
      CI    R0,0             IS SOUND DONE?
      JNE   HK1              NO- CONTINUE TO WAIT.
      RT                     RETURN TO CALLING ROUTINE.
```

PROGRAMMING HELP FILE:

The purpose of this column is to present techniques, and information that will be useful in the writing of programs for the TI-99/4A home computer. If you can provide some prgramming insight that might be useful to someone, please feel free to pass it on to me and I'll get it into the next newsletter.

BASIC/EX-BASIC:

This month I will discuss the use of the RANDOMIZE statement and the RND function.

In writing programs that play games or perform simulations one must usually incorporate an element of unpredictability. A game which plays the same every time quickly loses its challenge. Real life events are very rarely predictable. Can you predict the weather, or when your car will break down? Statistics use probabilities of an event, determined by past outcomes of the event, to predict the future behavior of an event. The prediction of the behavior of an event is usually stated in terms of the percentage of past events the expected event has happened. In all likely-hood if an event has occured 30 % of the time in the past it will continue to occur 30 % of the time in the future. Thus an event is said to have a 30 % chance of occurring in the near future.

To generate the occurence of an unpredictable event a random number generator is used. The XBASIC RND function generates a random number less than one but greater than or equal to zero. ie. 0 - .9999... RND can be used to generate the outcome of an event being simulated. If an event has a 30 % likely hood of happening the programmer can get a random value and compare it to .3. If the random value is less than or equal to .3 then the predicted event is said to have happened. If the random value is greater than .3 then the event is not said to have happened.

For example:

The probability of rain is said to be 50 % on a cloudy day. A portion of a program to simulate weather may look like this:

```
100 CLOUDY=-1
200 RAIN=( CLOUDY AND ( RND < 0.5 ) )
300 IF RAIN THEN PRINT "IT IS CLOUDY AND RAINING"
400 IF CLOUDY AND NOT RAIN THEN PRINT "IT IS CLOUDY AND NOT RAINING"
500 IF NOT CLOUDY THEN PRINT "SUNNY WEATHER TODAY"
```

How does a computer get a random number? A random number is simply a series of math transformations involving the digits of a large seed number. The transformations typically involve dividing the seed number by a large prime number and then dividing the remainder by a power of ten to get a decimal fraction. To generate the next random number the remainder of the prime number division is multiplied by a second prime number and is used as a new seed for the next random number. Because the old seed is used to generate a new seed the random numbers generated will repeat if you restore the seed to its initial value. In this way you will be able to get the random numbers generated to start over.

To change the initial seed of the TI XBasic random number generator the RANDOMIZE statement is used. To get a random seed value use RANDOMIZE by its self. To set the seed to a specific number place the number after RANDOMIZE. ie- RANDOMIZE 32767.

The numbers generated by the RND function are not truly random since they may be started over by restoring the initial seed value; however, they are often goodenough for the programming applications discussed.

ASSEMBLY:

There are no assembly equivalent to TI-X/Basic's IF-THEN-ELSE or FOR-NEXT stements, instead branching by the use of JUMP statements is used.

To use the JUMP statements properly they must follow statements which set the status register bits. The status register is where the status of the result of a data operation is placed. The status of an data operation is determined by comparing two data values. Depending if a data value is less than, greater than or equal to another value appropriate status bits will be set in the status register. Error in an operation will also be signaled in the overflow, carry, or parity status bits. When a JUMP instruction is executed the status register is be checked to determine if the JUMP is to be taken. If no JUMP is to be made the JUMP instruction is ignored and the next instruction is executed. Because of this condition checking a JUMP instruction is similar to an IF-THEN statement (ie- IF TRUE THEN 100).

A JUMP statement consists of a specific JUMP mnemonic and a destination label. The label in this statement must be within a +256 and -254 byte range from the location of the instruction. The label must be within this range because of the internal and relative nature of the JUMP instructions. The JUMP mnemonics reflect the nature of the condition that must exist before the jump will occur. For example: JOC label - will cause a jump to 'label' when the carry bit in the status register is set; otherwise, the next instruction is executed. Other JUMP mnemonics are:

JEQ ... Jump if equal.
JNE ... Jump if not equal.
JGT ... Jump if greater than.
JLT ... Jump if less than.
JOP ... Jump if odd parity.
JNC ... Jump if no carry.
JNO ... Jump if no overflow.
JMP ... Jump without regard to status register.

By the use of the various JUMP instructions, labels, and instructions which affect the status register, looping and IF-THEN-ELSE type operations may be performed easily in assembly language programs. Happy programming!

FORTH:

In TI-Basic you would use a FOR/NEXT statements to perform an iterative loop, in Forth there is a similar statement. This discussion explains how the Forth staement is written.

An example of a Forth iterative loop is as follows:

```
: EXAMPLE 10 0 DO SOMETHING LOOP ;
```

This is an example of a user defined Forth word which contains a loop. This loop will execute the word 'SOMETHING' exactly ten times. The starting value of the index is 0, then limiting value is 10. These values are taken from the stack before the loop is executed. The words between the keywords DO and LOOP will be executed each time the loop repeats. The loop will execute once before it compares the index to the limit. If the the index value is LESS THAN the limiting value it is incremented by one, and the loop will repeat. At the end of each iteration the index is again tested. If the index is equal the limit value the loop will end and exection will continue with any words following LOOP.

To alter the step value of the index you would change the loop to look like the following:

```
10 0 DO SOMETHING 2 +LOOP
```

Notice that LOOP has become +LOOP. This indicates that the step size is to be taken from the stack. In the case of the example the stepsize is 2. The step value must be place immediately before the +LOOP keyword. To step in a negative direction use a negative step value. There are other looping key words that are part of Forth but I will leave the discovery of those to the enthusiastic reader.

CURIOSITIES AND PASTIMES

This column features a monthly BRAIN TWISTER for your intellectual entertainment. This month's puzzle is called "Gambler's Payoff".

Seven men sat at a gambling table and agreed that whenever a player lost a game, he would double the money of each of the other players - that is, he was to give the other players each as much money as they then had in their pockets.

They played seven games and each lost a game in turn. Oddly, when they had finished, each man had exactly the same amount - $32 - in his pocket.

How much money did each man have before he sat down to play?
(Hint: if you write a program to solve this you should use an array.)

SOLUTION TO: "The Monkey and the Pulley."
We find the age of the monkey to be 1 1/2 years, and the mother to be 2 1/2 years, the monkey therefore weighing 2 1/2 lbs., and the weight the same. The rope weighed 1 1/4 lbs., or 20 oz.; and, as a foot of rope weighed 4 oz., the length of the rope was 5 ft.

```
100 CALL CLEAR
105 PRINT "STEPPING SOUNDS"::::::
110 FOR X=500 TO 1000 STEP 10
120 CALL SOUND(50,X,0,X-300,0,X+300,0)
130 NEXT X
140 CALL SOUND(1000,110,30)
150 FOR X=130 TO 1000 STEP 30
160 CALL SOUND(50,X,0,X*1.26,0,(X*1.26)*1.26,0)
170 NEXT X
180 CALL SOUND(1000,110,30)
190 GOTO 110
200 END
```

```
100 CALL CLEAR
110 DEF MOD56(N)=1+INT(N-56*INT(N/56))
120 DIM N(56)
130 READ NTS,SPD,VL
140 REP=4
150 ! REMOVE ! FROM 220 FOR OPTIONAL WAIT BETWEEN REPETITIONS.
160 WAIT=1
170 CNT=0
180 PRINT "MEMPHIS CITY BLUES"
190 PRINT : : : :"WPG 99/4A USER'S GROUP"
200 PRINT : : : : : : : : : :""
210 FOR I=1 TO NTS
220 READ N(I)
230 CALL SOUND(SPD,N(I),VL)
240 NEXT I
250 !CALL SOUND(WAIT,110,30)
260 CNT=CNT+1
270 IF CNT>REP THEN END
280 FOR I=1 TO NTS
290 CALL SOUND(SPD,N(I),VL,N(MOD56(I+28)),VL)
300 NEXT I
310 GOTO 250
320 DATA 56,330,0
330 DATA 165,208,247,277,294,277,247,208
340 DATA 220,277,330,370,392,370,330,277
350 DATA 165,208,247,277,294,277,247,208
360 DATA 247,311,370,415,440,415,370,311
370 DATA 220,277,330,370,392,370,330,277
380 DATA 165,208,220,233,247,233,220,208
390 DATA 165,208,220,233,247,233,220,208
```

5

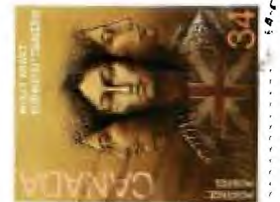
MICROpendium

CONVERT BASIC UPDATE

A number of readers have reported difficulty with th "Convert BASIC to Extended BASIC" User Note, whic appeared in the February 1986 issue. Anyone who does nc have a card which supports POKE will probably not be abl to get the program to run. We ran it using a CorComp Dis Controller Card, which supports POKE. The TI Disk Con troller card does not.

—JI

13



WINNIPEG, MA

TICR's Edmonton
PO BOX 11983
Edmonton Alberta
T5J 3L1

NEWSLETTER EDITOR
MUPG 99/4 USERS GROUP
P.O.B. 1715
WINNIPEG, MANITOBA
CANADA R3C 2Z6