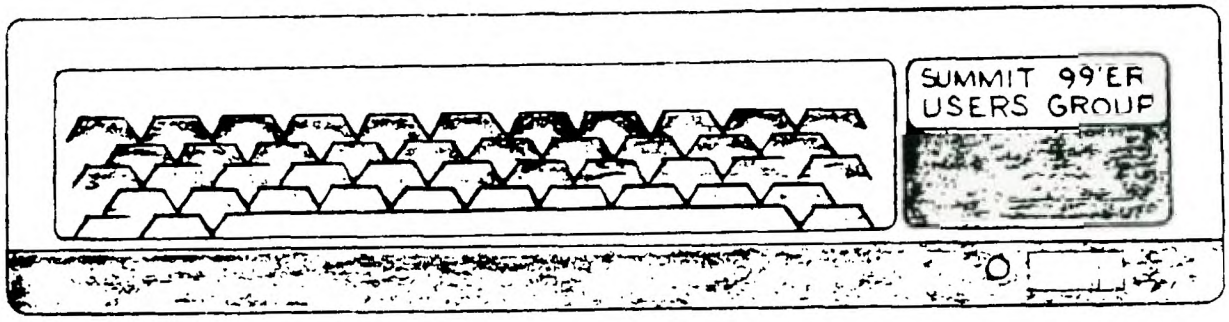
8407 (041)



JULY 1984 Vol. 2 No. 7

The July meeting will be held on Thursday, July 19 at Cuyahoga Falls High School at the corner of Fourth and Stow Streets in Room 413 - Physic's Lab. The August meeting will be held on August 16. Our meetings are always held on the third Thursday. Please be sure to sign in.

We will have a class on Basic for beginners. Please bring your Blue book that came with your keyboard.

This month's program will be on printers. We have lined up some people to come and demonstrate their printers.

Bert Haase called to say Rex's Salvage on Arlington has TI joysticks in the original boxes for $6.88.

RAFFLE

DATE: September Meeting
PLACE: Cuyahoga Falls High School Rm 413
TIME: 8 PM

1st prize= Extended Basic Module
2nd Prize= TI 99 4/A Console
3rd Prize= Parsec Game Module
4th Prize= Pair of Joysticks
5th Prize= Cassette Interface Cable

TICKETS - $1.50 each or 5 for $5.00
Ticket sales will start at the July 17 meeting at 7PM
Tickets can be purchased between meetings by calling Marilyn Bowen at 920-1884. Members may sign out up to 100 tickets for sale. Tickets and money must be turned in not later than 7:45PM September 20th. Members are responsible for lost tickets or unsold tickets not turned in by the deadline.

PRESIDENTS CORNER

For those of you who do not subscribe to the Home Computer Magazine I strongly recommend that you purchase the August, 1984 issue. This issue has many new hardware and software items as well as many new venders. For those of you trying to decide on a word processing system there is a good review of the Companion system by Judy Sanoian of the HCM staff. Those interested in making your own cassette cables Peter Bloch has written a article on how to make the cables complete with part numbers of the components that can be purchased from Radio Shack. A review of the popular burgertime game by the HCM staff plus many other interesting articles.

In the past two months our library has grown far beyond expectations thanks to the hard work of the library committe and the sharing of programs from other user groups. We need additional funds to purchase more blank tapes, disks and other items. Thanks to our new Vice President Norman Sorkin we will be able to raffle the following items. Extended basic module, TI 99 4/A console, a Parsec module, Joysticks, and a cassette cable. We urge each of you to support this effort to expand our library while at the same time having the opportunity to get one of these items at a bargin price.

NEW COMPUTER STORE IN RAVENA

The Computer Store opened in Ravena last week and will support third party software tor the TI 99 4/A. They also have Epson and Gemini printers at competive prices. The store is operated by Mr. Paul Kossick.
THE COMPUTER STORE
329 NORTH CHESTNUT
RAVENA, OHIO 44266
DAY PHONE 216-296-5981
NITE PHONE 216-296-2282

Micro-Biz has sent us a list of the software that they offer. Here is the list: Savings Bond Manager, C/D; Mailing Address/Label, C/D; Personal Record Keeping, C/D; Bingo Generator, C/D; Mr. Jumbo C/D; Jet Interceptor, C/D; Gongo, C/D; Color the Tune, C/D; Pacant, C/D; Stop, C/D; TV Pattern Generator, C/D; Easy Key, C/D; ABCABC, C/D; Catalog, D; Purchase Order/Invoice Blanks, C/D. C stands for cassette and D stands for disk. For more details on what the software is supposed to do, see the bulletin board at the meeting. They are offering discount rates for group orders. Contact Norm Sorkin if you are interested in any of their software.

Another software group has sent us their catalog. They are Intellestar. Their catalog will be posted on the bulletin board. They are offering group discount rates. If you are interested in any of their software, let us know?

We have recieved TI Assembly De-bug from TI. It is on disk and is available to members from the library.

This article comes to us from THE HUGGERS HOOSIER USERS GROUP, June 1984.

BEST OF THE NEWSLETTER!

THE HEART AND SOUL OF PERSONAL RECORD KEEPING

by Don Donlan (Part II)

The HEADER subprogram which resides in the Personal Record Keeping command module provides access to a "data dictionary". This dictionary defines the data that has been created by the PRK module. As I mentioned in the last article there are two formats or ways to code the "call" for this subprogram. For numeric information use [CALL H(R/W,INFO,FLD,V)] and to read/write character information use the format [CALL H(R/W,INFO,FLD,V$)]. If the "R/W" variable is "1" the subprogram will "read" information from the Header record. If the "R/W" variable is "0" (zero), the subprogram writes information to the Header record. "V" and "V$" are used, depending on which kind of information you wish to retrieve (V for numeric; V$ for character). The "FLD" variable is sometimes ignored, sometimes required, depending on what number is in the "INFO" variable. "INFO" is a number between 1 and 14 which determines what Header record information you will read or write. A "FLD" number is ignored for the first 8 kinds of information; the rest require a "FLD" number. The fourteen kinds of information stored by the Header record are as follows:

1 = File Name up to 10 characters long.
2 = Day of the month (a number from 1 to 31).
3 = Month (a number from 1 to 12).
4 = Year (a number from 0 to 99).
5 = Number of fields per record (a very important number because it will determine how many times we will need to ask for information found for the information types 9 thru 14 shown below). This number is automatically incremented each time a new "highest numbered" field is defined
6 = Number of records in this PRK file (also automatically incremented each time a new "highest numbered record" is written).
7 = Size of Header record (length is automatically calculated and entered).
8 = Size of the Data record (this too is automatically calculated and stored).

Now we come to the Header record information that describes the individual fields within each data record. This sequence of information is repeated for each of the fields, up to the number of fields indicated in Header record information 5 (see above). If "FLD" is 1, the first Data record is defined; if 2, the second; if 3, the third and so on.

9 = Name of the data field (up to 10 characters long).
10 = Type of Data field: 1=character 2=integer 3=decimal 4=Exponential.
11 = Size of Data field. This depends on the type:
Character data fields are 1 to 15 bytes long.
Integer data fields are 1 to 10 bytes long.
Decimal data fields are 2 to 11 bytes long.
Exponential or scientific notation fields are 8 to 13 bytes long.
12 = Number of decimal places. For character and integer data, this is zero.
For decimal data, the number would range from 1 to "size" minus 1.
And for Exponential or scientific notation the number is 0 to 5.
13 = Amount of space required for the Data field (as set by Header subprogram).
14 = Position of this field within Data record (as set by Header

As you can see, with this kind of information you can reveal the data base structure of this PRK file. I don't know if you could use a BASIC program and the Header subprogram to write your own data base structure. It would be interesting to see what would happen. A typical Header record might look like this:

MYRECORDS_10_22_83_2_15_50_30_NAME_1_15_0_15_1_PHONE NO._1_15_0_15_16

I use the "_" character to set the "INFO" fields apart. Above would be header record for a file called "MYRECORDS" which was last used October 22, 1983. It says there are 2 fields for every Data record, 15 such records in the file. The length of the header record itself is 50 bytes; the size of each data record is 30 bytes. The name of the first of our two fields is "NAME", which will have up to 15 characters of information in it, so we'll reserve 15 bytes. This first field starts in position 1 of the Data record. Our second sample field is called "PHONE NO.", again a 15 byte character field that needs up to 15 positons in the record, so we'll start it in position 16. The sample Header record above could be the start of a phone and telephone number data base that was created by the PRK command module. If you have the PRK module you might want go ahead and make up such a sample file and store it on disk for use by the programs that we will be presenting at the end of these articles. That way you will be able to 'test things out' on your own.

Next month we will review the GETPUT subprogram, the utility that reads and writes the Data records created/retreived by the PRK command module.

This "Best of the Newsletter" appeared in the November, 1983 issue of the HUGger Newsletter.

I would like to thank Rich and Ian and of course Pat for their articles. If you have an article that is news-worthy, I would like to include it in the next newsletter. The deadline for August is August 4.

Kathi Anderson, Editor

This article comes to us from Ian Mariano, one of our members.

ASSEMBLY LANGUAGE PROGRAMMING TIP

Here are some tips on good assembly language programming:

1.....STUDY ALL MNEMONIC INSTRUCTIONS
2.....MEMORIZE
3.....FIND SHORTCUTS
4.....FLOWCHART PROGRAMS
5.....TRANSLATE FLOWCHART INTO ASSEMBLY SOURCE CODE
6.....TEST AND MODIFY THE PROGRAM

These steps will help you to program assembly with greater potential. I will discuss each step with you in this article and by the time you are done reading it, I hope that you will gain a better grasp on programming assembly language.

STEP ONE:STUDY ALL MNEMONIC INSTRUCTIONS

Gaining an understanding of the instructions will help you create all kinds of routines that will help you make the program. Start from the beginning of the Editor/Assembler manual and read through it, try some of the application programs in it, that will help you understand what each instruction does. Always keep the manual somewhere where you can access it for reference, place markers where each section begins and where any important material is so you can just directly access the section you need to.

STEP TWO:MEMORIZE

As you learn the mnemonic instructions, memorize them, for if you do, you won't have to flip through the manual to find the format that the instruction takes. Also, memorization will allow you to program faster and more accurately in assembly language.

STEP THREE:FIND SHORTCUTS

When you develop a routine for a program, such as clearing a screen image table or clearing the screen, check over the source code and find anyplace in it where you can make the routine shorter and therfore save memory or time. For example, if a screen image table starts at >1800 and you must initialize >00 to >FF three times, your routine would probably look like this:TIME BYTE >00

```
PLACE  BYTE >00
STABLE EQU >1800
FOR    BYTE >04
LOOPS  LI R2,>06
       BLWP @VSBW
LOOP   LI R0,PLACE
       INC PLACE
       CB @PLACE,@>FF
       JNE LOOP
LOOP1  DEC PLACE
       CB @PLACE,@>00
       JNE LOOP1
       INC TIME
       CB @TIME,@FOR
       JNE LOOPS
       RT
```

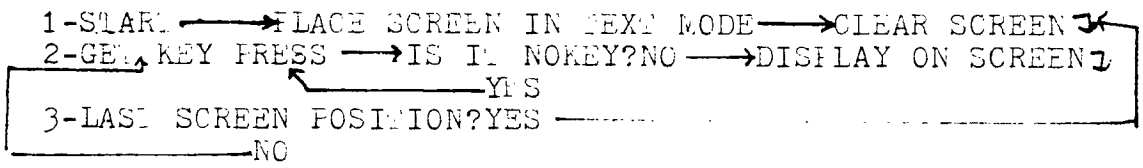
You want to speed it up, so make it into:

```
MAX    DATA >0100
TIME   BYTE >00
PLACE  BYTE >00
FOR    BYTE >04
STABLE LI R2,>06
       BLWP @VSBW
LOOP   LI R0,PLACE
       INC PLACE
       CB @PLACE,@MAX
       JNE LOOP
       INC TIME
       CB @TIME,@FOR
       JEQ RETURN
DECPLA DEC PLACE
       CB @PLACE,>00
       JNE DECPLA
       JMP STABLE
RETURN RT
```

If speed is what you want, use example two, for byte saver, use number one, see a choice in there?

STEP 4 &5:FLOWCHART PROGRAMS&TRANSLATE INTO SOURCE CODE

Decide what your program is to do, write it down as steps that the program will have to take in order to get the job done, this "diagram" of some sorts tells you what order to put your routines of the program in, or simply, a flowchart. Suppose we need a program to accept screen input for the entire screen, but it is only a test to see if we are advanced enough to do it, here is the flowchart:



Here we have a flowchart, now we must translate it into source code. Let's find what variables we need first,maximum screen positions: 960, current position:0,key ascii:>8375,nokey pressed:>FF,incriment: 1. Let's see, ah yes, here is the first part and second part now completed, all we have to do is put it into source code, which could be:

```
        DEF SCREEN
        REF VMBW,KSCAN,VWTR
UNIT    BYTE >00        Scan entire keyboard
FIRE    EQU >8375       ASCII value of key pressed
NOK     BYTE >FF        No key pressed
POSI    DATA 0          Current screen position
MAXP    DATA 961        Maximum screen position+1for all positions
ONE     DATA 1          Increment or decrement
SCREEN  LI R0,>01F0     Initialize text
        BLWP @VWTR      mode
        MOVB @>8374,@UNIT
CLEAR   LI R0,POSI
        LI R1,>20       Space character
        LI R2,1
        BLWP @VMBW
        A @POSI,@ONE
        C @POSI,@MAXP   All screen cleared?
        JNE CLEAR       If not, continue
LOOP    S @POSI,@ONE
        C @POSI,0
        JNE LOOP
GETKEY  BLWP @KSCAN
        CB @FIRE,@NOK   No key pressed?
```

```
        JEQ GETKEY  Yes, scan again
DISPLY  LI R0,POSI
        ~~LI R1,FIRE~~ MOVB @FIRE,R1
        LI R2,1
        BLWP @VMBW  Display ASCII char
        A @POSI,@ONE
        C @POSI,@MAXI Max position on screen?
        JEQ CLEAR   Yes, clear screen
        JMP GETKEY  No, Continue
        END
```

With flowcharting, writing the program is a lot easier for you to do.

STEP SIX: TEST AND MODIFY THE PROGRAM

Now that the program is complete, you must check for any "bugs" in the program. Now you can add any cosmetic changes if you wish. If at all possible, use the DEBUG program that came with your E/A. With DEBUG you can edit the program while it is in use.

THE END

This article comes to us from Rich and Doris Williams, who are members of our group.

BIO-RHYTHMS

The theory behind Bio-Rhythms is that our bodies produce, store up and release energy in regular cycles. These cycles are started at birth and never gain or lose a second; maintaining perfect time; during our lifetime.

Bio-Rhythms are measured in three different cycles: a twenty-three day physical that governs your physical strength, endurance, energy, confidence and sex drive; a twenty-eight day emotional cycle that governs mood, sensitivity, creativity and friendliness. The last cycle is a thirty-three day mental cycle that governs learning ability, logic and memory.

The Bio-Rhythms are charted as low cycle, high cycle and caution days. When a cycle is high, these are good days for you and your outlooks are good. When a cycle is low, your outlook is negative and depressed and these are not good days. Caution days occur when a cycle crosses the midline. These days mean your system is in a state of flux as it switches over. On these days, you can be more accident and error prone. It is even more important to watch for double or triple caution days when two or more cycles cross on the same day. Take extra care on these days.

The physical cycle on the high side means you feel stronger, healthier and energy is flowing through your system. The low side means your energy flow has become a trickle, you are less energentic, less confident, more prone to illness and tire more easily. Physical caution days cause your energy flow to shut down; timing, reflexes and perceptions are dulled and you are, statistically, up to five times more accident prone.

The high side of the emotional cycle makes you more optomistic, cheerful, affectionate and your creative processes are stimulated. The low side causes you to be moody, nervous and less creative. Emotional caution days can be very upsetting and when in conjunction with a physical caution day can cause unsettling problems for some.

The mental cycle high side means your learning ability, logic and memory are operating at peak efficiency. The low side means your analysing perspective and grasp of new ideas are reduced. Mental caution days are not a potent on the system as other caution days but important decisions and major scholastic exams should be avoided on these days.

When you run this program, the first thing to come up after the misspelled title is the question of whether or not a thermal printer is attached to your system. A "yes" answer will enable an output for printing the chart.

The next inputs are the subjects name and then said subjects date of birth divided into month (MM), day (DD) and year (YY(YY)). Year designations are set up to accomodate

two or four digits (82 or 1982). [One important note for year abbreviation to two digits; the program will conclude you were born in the twentieth century.] All dates must be separated by commas. These notes also apply for the next input statement of the current date.

Once these dates have been entered, the program will calculate the total number of days the subject has lived and an approximation of the total years. From this, the program will create the Bio- Rhythm chart starting with the current date entered.

The evaluation of the chart is as follows: the red "P" stands for the physical cycle; green "I" stands for the mental (intellectual) cycle and the yellow "E" stands for the emotional cycle. A star "*" is to show an overlapping of cycles and does not signify critical days. The chart is divided into a minus(-) for the low side, a plus(+) for the high side of the cycle and a zero(0) to indicate the critical line.

Pressing the space bar will continue to run the day to day chart until function 'clear' is used to break in.

When compared with a mathmatically calculated chart, this program is comparatively accurate.

For a "Basic" program, it seems to be well-written and the graphics are suited to the program.

By Rich and Doris Williams

This program will be available from the library at the coming meeting.

LIST OF BOARD MEMBERS AND THEIR HOME PHONE NUMBERS

President, Pat Bowen	920-1884
Vice President, Norm Sorkin	678-2560
Librarian, Leroy Martin	666-3984
V.P. Program, John Tuesday	
Secretary,	
Treasurer, Betty Duncan	633-5217
Educational Director, John Curry	929-8824
Editor, Kathi Anderson	923-7530

This article comes to us from the PENN OHIO 99/4A HOME COMPUTER USERS GROUP newsletter, June 1984.

FOR BEGINNERS
RANDOM NUMBERS

This is for those just learning to program. Many times, especially with games, or programs being written to teach kids, one wants an element of randomness. You have a very powerful function available, RND. This command is used as a numeric variable in your program statements. To non mathematicians it is a bit confusing as to what happens, but in a minute you will see the extreme versatility we have with this command. A random sequence of numbers is a very elusive thing. The TI 99/4A RND command actually uses a routine that generates a pseudo-random sequence of numbers. Each time you run the program you will get the same random sequence.

To help make this more truly random, you have two simple options. At the beginning, or even multiple times throughout a program, use as a command in a line of the program, RANDOMIZE or RANDOMIZE followed by a number, or any numeric expression. This number is called a seed. Note that only the first two bytes of the seed are used. If you follow RANDOMIZE with a seed, you will get the same random sequence each time you run the program if the first two bytes of the seed are the same. If you do not use a seed you will get a truly random and different sequence of numbers each time you run the program.

Now, back to what RND actually does. This command generates and returns a random number that is equal to or greater than 0 and less than 1. When I read that for the first time in the user's guide, I thought good grief, that sure sounds worthless. What you discover though when you use the command that they left out some very valuable information. This number is actually a 10 digit number, which means that simply by multipying RND by 10,000,000,000 you can generate a random number between 0 and 10 billion. Of course you can multiply it by any number or add any number to it, etc, so that you are limited only by your imagination in creating random numbers to fit any need. Usually you want an integer, that is you don't want all those digits to the right of the decimal point. Use the INT function. They do give you a formula in the users guide, which I will repeat below, but it is much nicer to understand why the formula works, because then you are not limited by it.

To get a random integer between and including A and B, where A<B, do the following:

```
10 RANDOMIZE
20 C=INT((B-A+1)*RND)+A
```

Adding A at the end sets your lower limit, as the rest of the expression generates a number either 0 or greater. A is subtracted from B in the portion that is multiplied by RND because otherwise your final number will exceed your upper limit as A is added back on at the end. 1 is added to the multiplier so that the end result will include the upper limit. That is because the INT function drops the digits to the right of the decimal point for positive numbers, and for negative numbers returns the next lowest integer (INT(-2.3)=-3).

Frank Krautter

SUMMIT'99ers USERS GROUP
Kathi Anderson, Editor
3240 Bailey Road
Cuyahoga Falls, Ohio 44221