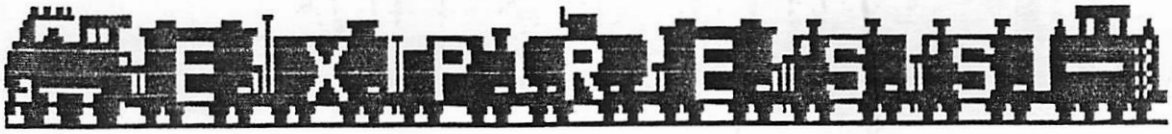


The Official Publication of The

PENN-JERSEY COLOR COMPUTER CLUB

MAYBROOK BBS

215-760-0456

JUNE 1992

NEW MEETING ROOM FOR PJCCC

Rick Hengeveld

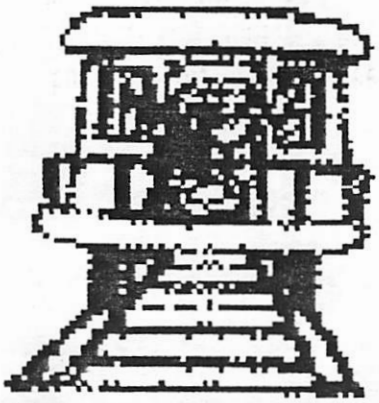
The PJCCC was never a club that could be considered "At a standstill" Seems that the PJCCC is on the move again! This time thanks to Richard Kravitz our move will be to a new meeting area. Tentatively we will be moving to room 190 in the main college hall. Many thanks to Richard for all the leg and contact work he's put in to secure our meeting areas.

Coming up at the June meeting, We'll see Clyde Gano demo the "Simply Better" word processor. The system looks like a winner!

Welcome back to Pete Unks! After a bout of pneumonia with complications, Pete is back on duty with the 6809 Express.

Remember the PJCCC's main goal is to help and support its members. So if your having any problems or questions, feel free to ask them at our next meeting.

IF YOU MISSED OUR LAST MEETING YOU MISSED A GREAT TELECOMMUNICATIONS DEMO!



THE LIBRARY CAR

Alan J. Wagner

Welcome once again to the Library Car. Make sure you pull up a comfortable seat this month as we're going to be getting into trying to figure out just what we're going to do with our project program. As you may know I doubled as editor last month, so I didn't get quite as far as I would have liked to have been by this time, but I'll try to make this as informative as ever. Last time I asked you to think about how you might go about setting up the program, but to do it in english not Basic09. I started my cogitation on the subject and found it easiest for me to write an outline. I am not going to try to include the entire outline in the Express, but rest assured that the entire program WILL make it to these pages. What I will include is enough of the outline so you can see how I am going about designing the program. The following is some of that outline and the stream of thought notes I made to myself.

Yes, I did the notes and outlines on a word processor. I find it easier to just plop things down on the crt and arrange them in a logical order later. Note that the section labeled Menu outline is just that an outline for the menu that would appear upon startup. There are no real details as to how these things will be actually encoded in Basic09, as well there should not be at this time. Here we are just trying to get a general feel for where we want the program to go and what features we feel should be included.

****

Main program should be just menu to sub routines.

Title page with copyright to myself and club should appear before first menu. Make it a press key to continue.

Menu outline

I. Create new database
II. Select existing base
    1. Add data
    2. Update data
        a. Search by field
            1)Display record
            2)Change record
        b. Sort by field
    3. View data
        a. Search by field
            1)Print on screen
            2)Print on paper
        b. Sort by field
    4. Delete data
        NOTE: Must have double check against accidents
III. Delete entire database
        NOTE: Must have double

check against
accidents

IV. Exit program.

Create database would create a directory using user given database name. Directory will contain the main record file and an index file for each field by which searches will be allowed.

Similar items under II.2 and II.3 will call same subroutine. Must include flag to indicate what to do with record once found, depending on calling routine. This would branch to another subroutine to either just display or also allow changes. Add data could create a blank record and then call the update routine.

Only record indices for specified fields will be sorted during any given access to the sort. Flags must be set in all indices when an add data is done. A flag must be set in only those field indices effected when an update occurs. Perhaps a separate file for the flags would be appropriate so that only one file would have to be opened to update the effected index flags.

****

The above is only my first draft of the menu section of the program and will likely have at least a few changes as we go along. Not everything in these notes will be included and some things not there will appear. This is all part of the process. Even now as I am assembling this into the article, I noticed some things that weren't as clear I as I felt they should be, so I rearranged some of it. Notice too how I made notes concerning subroutines and possible variables as the ideas occurred to me. If you don't get them recorded as they occur, you'll forget them by the time it comes around to including them in the program.

Only the roman numeral headings will be in the "main" program. All the rest of the items will be in "subroutine" programs called from the "main" program. This is called modular programming. The big advantage of this is that if one module needs to be changed for whatever reason, it is very likely that only that module will need be changed and not the whole program. Also once we write the modules, we may find that we can use them in other programs with little or no changes. We just call them from the new program. In addition, we can build the main menu program and test it into dummy routines long before we have to write a file access routine or have a file for the records.

Each step of the way we can write and test before having to write a complex subroutine. This makes the debugging of the program easier. This is known as top down programming. There is also a method known as bottom up programming where one starts with the most elementary of the subroutines and them builds back up towards the main control program. The problem with this is that you don't always know what you need at that very bottom level until you have written some of the top. The advantage is that you don't need to write dummy subroutines to test the higher level routines. For this series we are going to use the top down approach.

Let's try our hand at the encoding of the primary menu section. Below is how I

attempted to fulfill the requirements set out above.

```
PROCEDURE inventory
  0000      REM **********************
    001A                        REM    *
     *
  0034          REM  *         BASIC09
     *
    004E                        REM    *
     *
  006       REM *  Home Inventory      *
    0083                        REM    *
     *
  009D         REM * Copyright May 1992
*
    00B7                        REM    *
     *
  00D1       REM * Alan J. Wagner, Sr. *
  00EB         REM *  222 Jefferson Ct.
*
  0105         REM *    Quakertown, PA
 *
   011F          REM *               and
   *
    0139                        REM    *
     *
   0153          REM *   The PennJersey
 *
    016D                        REM    *
     *
   0187          REM *    Color Computer
*
    01A1                        REM    *
     *
   01BB           REM *             Club
     *
    01D5                        REM    *
     *
  01EF      REM **********************
  0209        REM
  020C         REM Main Menu
  0218        REM
  021B        DIM ans:STRING[1]
  0227        DIM x:INTEGER
  022E        DIM leave:BOOLEAN
  0235        leave:=FALSE
  023B        PRINT CHR$(12)
  0240        RUN PRINTAT(30,5)
  024B        PRINT "Inventory Program"
  0260        RUN PRINTAT(30,7)
  026B        PRINT "Copyright May 1992"
  0281        RUN PRINTAT(30,9)
  028C        PRINT "Alan J. Wagner, Sr."
  02A3        RUN PRINTAT(31,10)
  02AE        PRINT "222 Jefferson Ct."
  02C3        RUN PRINTAT(32,11)
  02CE        PRINT "Quakertown, PA"
  02E0        RUN PRINTAT(37,13)
  02EB        PRINT "and"
  02F2        RUN PRINTAT(32,15)
  02FD        PRINT "The PennJersey"
  030F        RUN PRINTAT(32,16)
  031A        PRINT "Color Computer"
  032C        RUN PRINTAT(37,17)
  0337        PRINT "Club"
  033F        RUN PRINTAT(28,23)
   034A          PRINT "Press ENTER to
continue";
  0366        ans:=""
  036D        INPUT ans
  0372        LOOP
  0374          PRINT CHR$(12)
  0379          RUN PRINTAT(30,5)
  0384          PRINT "Main Menu"
  0391          RUN PRINTAT(25,10)
   039C           PRINT "1)  Create New
Database"
  03B7          RUN PRINTAT(25,12)
   03C2               PRINT "2)  Select
Existing Database"
  03E2          RUN PRINTAT(25,14)
   03ED             PRINT "3)  Delete a
Database"
  0406          RUN PRINTAT(25,16)
  0411          PRINT "4)  Exit Program"
```

```
  0425          ans:=""
   042C               WHILE ans<"1" OR
ans>"4" DO
  0441             RUN PRINTAT(25,20)
   044C              PRINT "Enter selection
";
  0461             INPUT ans
   0466                   PRINT CHR$(9);
CHR$(3)
  046F           ENDWHILE
   0473              ON VAL(ans) GOSUB
10,20,30,40
  048C         EXITIF leave THEN
  0495           PRINT CHR$(12)
  049A         ENDEXIT
  049E         ENDLOOP
  04A2         END
  04A4  10     RUN makebase
  04AB         RETURN
  04AD  20     RUN existing
  04B4         RETURN
  04B6  30     RUN basedelete
  04BD         RETURN
  04BF  40     leave:=TRUE
  04C8         RETURN
```

ALL Basic09 programs start with the line "PROCEDURE 'name'". This is important to remember. If you create the procedure/program in the Basic09 editor, the addition of this line is automatic. But, if you use your favorite wordprocessor, you must remember to put this line in as the very first thing in the file. No spaces, carriage returns, or any other characters may precede this information or Basic09 won't recognize it as a valid procedure! As a reminder, the numbers at the beginning of each line are the memory offset, from the beginning of the procedure, of the first actual character of that line.

Now let's get into the commands. REM is the remark command. Anything that follows this command is ignored by the computer when the program is executed. These lines are put in there for the benefit of the humans that my look at the listing. They are useful to document a program, to leave yourself notes as to where a certain section is headed while you are still developing the program, and to record copyright information. DIM is the command to dimension a variable. We exercised this one over the last couple of sessions.

PRINT causes the program to place what follows on the path named or, as in all the cases in our example, on the default path, which is the terminal or crt. CHR$() causes the PRINT command to send the value in the parentheses down the selected path. This can be any number from 0 to 255. Some of these numbers represent the letters, numbers, and other characters on the keyboard. Others of these are non-printable signals or commands to the device at the end of the path. CHR$(12) is just such a command. This tells the terminal to clear the screen and position the cursor in the upper left of the screen. If we had sent this to a printer, most printers would have executed a form feed. In the section of your OS9 manual called "OS9 Commands", Appendix B, pages 4 & 5, have a listing of the "characters" that are effective on an alphanumeric screen.

Basic09 does not have a built-in command to locate the cursor on the screen as does RSDOS. What it does have is also in the above mentioned appendix. If you opened to that appendix, you may have noticed that if you send a "2" to the terminal,

you can then send two more numbers that represent the column and the row where you want the cursor to appear. I wrote a short program I called PRINTAT (after the RSDOS command). The reason I felt a program was necessary was that the column and row don't equate the way you and I think. You have to add hex 20 (decimal 32) to each number before sending it or the cursor won't be where you want it. It seemed easier to let the computer do the task. Once the program was written I no longer had to do the math and the call for positioning the cursor seemed more familiar.

This also demonstrates another important aspect of Basic09. You can call one program from within another. You can even pass variables back and forth and the calling program doesn't get blown away because the second program was called. The first program's variables remain intact as does the pointer to the location where execution was within the program so execution proceeds where it left off upon return. If recall I had spoken of modular programming. This is the key that allows us to do it!

```
PROCEDURE printat
  0000        PARAM x,y:INTEGER
  000B        DIM column,row:INTEGER
  0016        column=x+32
  0021        row=y+32
 002C   PRINT   CHR$(2); CHR$(column);
CHR$(row);
  003C        END
```

Notice the command PARAM. This acts very much the same as a DIM command but tells the program to expect these variables to be passed from a calling program. If you look at the menu program, notice how when I called printat I had two numbers in parentheses. These are the numbers printat takes to be x and y respectively. You are not limited to integer numbers, but can pass any variable type as long as you keep the types the same. Neither program is aware of what the variable is called in the other program.

So, you can use the same variable name in two programs, even if they don't represent the same thing or maybe even the same type. What is important is that the order in which the variables are sent and expected. If the first variable sent is an integer, the receiving program had better be looking for an integer in its first PARAM variable. If the receiving program is looking for a string as the fourth PARAM variable, the fourth variable in the parentheses of the call to that program had better be a string. If this arrangement is not adhered to, the program will assume it was sent the proper stuff and attempt to use what it gets. Needless to say, the results won't be what you're expecting and can not be predicted.

Notice in the PRINT command that there are three CHR$() commands connected by semi-colons. Semi-colons allow us to concatenate CHR$() commands so that they are sent in one continuous stream with no carriage returns in between. Notice also the semi-colon at the end of the line.
This tells the terminal to wait at that point for more characters to follow. If the semi-colon weren't there, the terminal would execute a carriage return putting the cursor at the beginning of the next line and defeating the whole purpose of sending

the characters in the first place.

You might think that calling an outside program would take a lot of time, but that just isn't so. If the programs are stored as separate files, the first time the second program is called, it will have to be called from disk. After that, however, it is in memory and can be called as quickly as a subroutine within the same program. If the programs are all stored in one file on the disk, which can easily be done, they are all loaded at once so there isn't even the wait on the first call.

You may have noticed that in the assignment statements such as "leave :=FALSE" and 'ans:=""' that I included a colon before the equals sign. This is not really needed but is recommended to differentiate between an assignment statement and and a test for equality.

INPUT accesses the path from the terminal and requests data from the terminal. Whatever is typed is then entered into the variable(s) that appear after the INPUT command. In this case the variable ans. If you look back at the beginning of the menu program the DIM statement for the ans variable dimensioned ans as being one character long. What happens if you type more than one character in reply to the INPUT command?
More than one letter will appear on the crt, but only the first is captured by the variable. The rest fall on the floor behind the computer. You always wondered where that pile of stuff came from. Well, now you know.

The LOOP command has a companion that you'll find near the end of this program. It is know as ENDLOOP. This pair define the beginning and end of a never ending loop. There are only two ways out of the loop. One is the group of commands EXITIF/THEN/ENDEXIT which appear near the end of the program. They work like an IF/THEN/ENDIF group. That is to say, EXITIF a statement is TRUE THEN do all the command statement lines up to the EXITIF statement. There doesn't have to be any lines between the THEN and the EXITIF or there can be as many as it takes to leave the loop in an orderly manner. In our case there is one statement which clears the screen. When the ENDEXIT statement is encountered, the program immediately goes to the ENDLOOP statement and executes whatever is the next statement. In this case the END of the program.

There is a WHILE/DO/ENDWHILE loop after we print the menu to check for an appropriate response. Basic09 has a number of looping statements that make programming easier. A WHILE loop continues to go around as long as the conditional part of the statement is TRUE. In this case, as long as ans is less than 1 or more than 4. Because the numbers are represented as a string, the comparison is by their ASCII value. This has the advantage in this case of also checking for letters and other keyboard characters without giving an error for type mismatch.

A WHILE loop checks for the condition at the top or beginning of the loop. This means that if the condition fails upon entering the loop, the loop is NOT executed even once. This is why I assigned a null to ans before the WHILE loop was encountered, to ensure that we

would enter the loop. There is a loop arrangement that checks for the condition at the bottom of the loop, but that is a topic for another time.

Immediately after the WHILE loop is an ON/GOSUB statement. ON the numerical value of a variable or statement that appears between the ON and the GOSUB, this statement selects the appropriate line number. In other words, if the varible or statement equate to a value of 1, the first line number is chosen. If the value is 3, the third line number is chosen. This is one of the very few places in Basic09 where line numbers are required. Once a line number is chosen the program branches to that point in the program and executes the series of statements until a RETURN is encountered. At that point execution of the program returns to the line directly after the line containing the GOSUB. In this case, we check to see if we have called for the program to terminate. If leave is TRUE then the program exits the LOOP and ENDs, otherwise it goes back to the top of the loop.

There is just one more thing I need to cover this session. We need to construct a dummy program the menu program can call to test the various menu item program calls. What follows is a dummy for the first menu item. You can write similar programs for items 2 and 3.

```
PROCEDURE makebase
  0000        DIM x:INTEGER
  0007        PRINT CHR$(12)
   000C             PRINT "HELLO FROM
MAKEBASE!"
  0024        FOR x=1 TO 4000
  0035        NEXT x
  0040        END
```

One point of interest before I leave you to chew on this 'til next time. Remember way back I mentioned that Basic09 was faster than RSDOS Basic? This little program makebase is a good example of that. Notice that even though we are counting to 4000, the time delay is not very long at all. If you write a little program similar to this in RSDOS Basic, you'll find that a loop that counts to 4000 takes a very noticeably longer time to execute.

For those of you who had the chance to see my computer die before your eyes during the demonstration at the last meeting, you'll be happy to know I got it back running the next day. It was fortunate that I had a spare 6809 laying around as that is what smoked. I think I may have moved the multipack enough that I shorted a couple of the lines that go directly to the cpu. That'll get ya every time. Well, 'til next time, may all your computing be enjoyable and all the smoke you see be from your bar-b-que.

The Maverick Report
Rick Hengeveld

The Maverick BBS has logged over 680 Calls. With a user base of about 40 people. Some of these forty users are inactive. I hope your not on that list!

Don't Miss
Clyde Gano's
demonstration of
SIMPLY BETTER
the
outstanding
(and inexpensive)
WORD PROCESSOR
that even
Rick Hengeveld
wants to see!

FRIDAY JUNE 27
7 pm

GET ONLINE WITH THE MAVERICK
BBSi DIAL 215-760-0456

6809 EXPRESS

The "6809 EXPRESS" is the official publication of the PENN-JERSEY COLOR COMPUTER CLUB. The club is based in the greater Lehigh Valley of Northeastern Pennsylvania including sections of Northwestern New Jersey. Any non-profit organization may reprint any part of this newsletter provided credit is given. PJCCC will gladly exchange newsletters with any other computer club. Send requests to EDITOR, 6809 EXPRESS, Penn-Jersey Color Computer Club, 145 Seventh Street, Phillipsburg, NJ 08865. PJCCC assumes no responsibility for errors or omissions. PJCCC assumes no liability for damages resulting from the use of any information or programs contained in this newsletter.

The Official Publication of The
PENN-JERSEY COLOR COMPUTER CLUB

H. Peter Unks, Editor

Eric Rhyder

FIRST CLASS MAIL