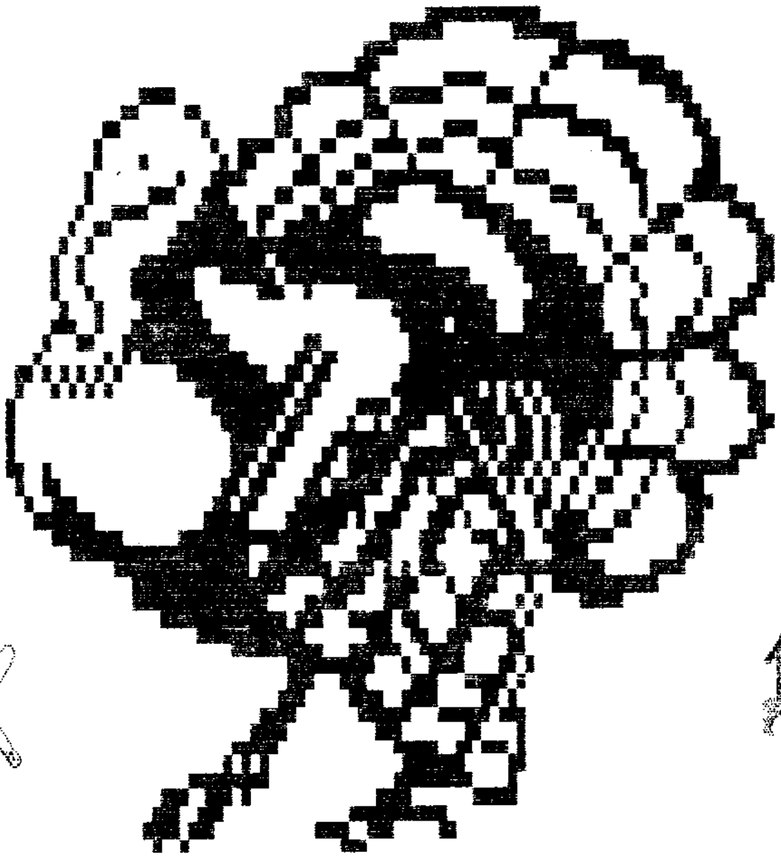
Guilford 99'ers
Happy Thanksgiving



Volume 6 Number 11 November 1989

Dues are $12.00 Per Family, Per Year
Send Check To: 3202 Canterbury Drive
Greensboro, N.C. 27408

Software Library is for dues paying members only

Happy Thanksgiving!

Guilford 99'ers

OUR NEXT MEETING

DATE: Nov 7, 1989 Time: 7:30 PM. Place: Glenwood Recreation Center, 2010 S. Chapman Street.

Program for this meeting will be a demonstration of the TI to IBM connection. For those of you who have a TI at home and use an IBM or a clone at work this will show you how to get the best of both worlds. Be sure to be there!!

MINUTES

The October meeting of the Guilford 99er Users' Group was held at the Glenwood Community Center on Chapman Street in Greensboro, N.C. on October 3, 1989. There were 7 members and 1 new member present. The Group welcoms Mr. Donald Ray into it's ranks.

The meeting was called to order at 7:30 P.M. by Vice President Emmett Hughes. Since there were no minutes of the September meeting due to the absence of the Secretary, only the Treasurer report.

OLD BUSINESS:

There was no old business to discuss.

NEW BUSINESS:

a. Under new business, the Vice President suggested to the Secretary that it would be a good idea to bring all of the exchange newsletters to the Center and keep them stored there so members would have a better chance to pick the ones they wanted to borrow. This will be done at the November meeting.

b. Herman Geschwind donated a COMPUTE! program book with 33 programs for the TI. This book is to be auctioned off at the November meeting to the highest bidder. The book is 200 pages long and covers from money management to games with much more in between. Try to come and make a bid on this well written book. The proceeds will go to the club treasury.

c. The nominating committee was suppose to be chosen, but due to the lack of a quorum it was suggested that the following be nominated: Bob Carmany for President, Emmett Hughes-Vice President, L.F. 'Mac' Jones-Secretary/Treasurer, George von Seth-Newsletter Editor, and Bill Woodruff-Library. There will be a chance for anyone to call a nomination from the floor at the November meeting when the official ballots will be cast.

After business, Bob Carmany gave a program on different ways to speed-up extended programs. Pre Scan It was demoed and also Bob showed how Craig Miller used Call Color to speed up parts of his Night Mission. He also showed some ways sprites are used to get three dimensional effects. A very good demo Bob. The meeting was adjourned at 9:00 P.M.

Respectfully submitted,
L.F. 'Mac' Jones, Sec/Treas.
Guilford 99er Users' Group

XB TUTORIALS PART 3

Our next example will be a good start on a non-trivial utility program for printing out TI BASIC or XB listings on a 80 column printer in two side by side columns which preserve the normal screen listing format. If you just LIST "RS232.BA=...." then the computer sends it out in DIS/VAR 80 format and it is up to you to tell the printer how to handle it. Something approaching screen image format is only obtained (with extra paper consumption) with the printer margins set way in. 80-col printout beats none at all by miles but let's try to be fancier. If you don't have disk or printer then this lesson won't be of immediate use, but will still be a good example to work through as a programming exercise. We might as well do something useful.

. First we figure out what needs to be done, and work out a set of procedures that can be CALLed as needed. The program ill do only the minimum necessary to do the job properly. Bells and whistles can be added later. In one or two places we will make provision for adding extras (bells and whistles have nothing on speech) by dummy subprograms which can be filled in later. For a good discussion of the use of such "stubs" see the excellent book by R. Mateosian, "Inside Basic Games". The detailed coding examples in this book are in Apple or Trash-80 Basics (does anyone remember any more what a TRS-80 was ?), but Mateosian develops ideas in a form much more in tune with a TI XB subprogram realisation than with these less capable Basics.

So let's start designing our program by deciding what we want it to do. We want the output nicely formatted on the page with top and bottom margins, in 2 columns each in screen image (28 char/line) format. More columns (assuming the output device will handle them) are no problem -- once you can count to 2 then 3 is easy. Lines of Basic are not to be split from from one column to the next or from one page to the next. Some things commonly encountered in printed listings, such as indenting of FOR-NEXT loops don't fit at all well with the multi-statement lines of XB (but might with TI Basic listings) so will not even be thought about here. On the other hand insertion of spaces before REM or SUB statements greatly improves the readability of XB listings, without doing violence to the idea of being screen list compatible. Page numbering is no big deal to add (a console only XB program can fill 6 pages).

At the other end of the business the LISTing to be printed is assumed taken from a disk file such as DSK1.LIST where it has been written by LIST "DSK1.LIST". A trivial difficulty easily taken care of is the blank first record written by LIST. The real problem is that LIST doesn't care about preserving XB lines as distinct entities. Each XB line starts out as a separate print record and if it is less than 80 characters long stays in one piece. XB lines can easily extend into 2 print records and more (Basic lines much less frequently), but LIST places no markers to show which print records contain the start of XB lines. So if we are going to meet our specification that XB lines be treated exactly as in a screen list then something more subtle than a simple LINPUT is needed. There's one of our most important building blocks identified --- SUB BASICLINE(...).

Any utility program needs title and advice screens so there's SUB TITLES to keep all the details from cluttering the main program. The program will also need SUB OPTIONS(...) to handle file and device name entry and print options which might be offered.

Now the real core of the program is the way in which it must assemble a whole page before printing anything because line feed moves ever on (at least it does on our TI-99 printer). So we need SUB PAGEBUFFER(....) to take the output of BASICLINES, chop it into screen format hunks and decide where these are to be located on the page. Then we need SUB PRINTPAGE(...) to massage the completed pages and ship them off to the printer. That about sums up the sub-programs that are called directly from the main program, and all that is necessary is to figure out the initialisation -- DIMs, default filenames etc etc, and to write the logic for program flow.

Before we start writing any code we should decide what utility sub-programs are to be used by those already defined. As the list is written into columns SUB WRITECOL(...) is a good candidate for repeated use, and SUB WRITEPAR(...) to take a line of BASIC and return it chopped up into 28 character lines to WRITECOL. Since BASICLINE fetches the input records it is the appropriate place to detect End Of File. We might as well use PRINTPAGE to wipe the slate clean before writing a new page.

Let's dress up the input of filenames and Yes/No responses a little as SUB FILENAME(...) and SUB YN(...) , with SUB MORE(...) to end it all. Other useful utility sub-programs which will be included are SUB TXTCOL(..) to change display colors in one CALL, SUB KEYCON to carry the burden of "press any key to continue", and SUB DELAY(..) is always handy.

That about finishes the roster of procedures necessary to make up the listing program, and now the detailed coding can start after some thought on the necessary chains of parameter passing. The principle that you should plan your programs from the top down and code them from the bottom up is just as valid in Extended Basic as it is in TI-LOGO or TI-FORTH where the form of the language makes it difficult to do otherwise. Sub-programs make it possible to go the same way in XB with ease. Less capable dialects of Basic make it a lot harder to keep your thoughts organised and your code on the rails.

The actual program will now be listed piece by piece and commented on in detail. The listing has been transferred into this TI-Writer file from a working copy of the program using a more elaborate version. The present program is actually a simplified version of the one originally written, but is powerful enough to do a useful job.

```
100 REM ** SIMPLIST **
110 REM * PRINTER LIST *
120 REM ** FROM DISK **
```

```
130 REM -FUNNELWEB FARM-
140 OPTION BASE 1 :: DIM PRLN$(66,2)
150 REM * DEFAULT VALUES *
160 CALL TITLES :: SFIL$="DSK1.LIST" :: PDEV$="RS232.BA=4800"
170 CALL KEYCON
```

The first part of the main program shown here sets default values and DIMensions the string array PRLN$ for two columns of 66 lines each. The top and bottom few lines will be left blank so that page format is obtained without sending printer control codes. A 66 line/page, 80 col. printer is assumed.

```
180 REM * New File Entry *
190 CALL OPTIONS(SFIL$,PDEV$):: ENDFILE=0 :: LINPUT #1:NEW$
200 REM * New Page Entry *
210 CALL PAGEBUFFER(PRLN$(,),ENDFILE)
220 CALL PRINTPAGE(PRLN$(,), PDEV$):: IF ENDFILE=0 THEN 210
230 REM * End OR Next *
240 CLOSE #1 :: CLOSE #2 ::CALL MORE(NM):: IF NM THEN CALL SPEAK("GOODBYE"):: GOTO 250 ELSE 190
250 STOP
```

OPTIONS returns file and device names as entered there, and the remainder of line 190 resets the End of File flag, and throws away the first line of the list-file. At new page entry the page buffer is filled and then printed out repeatedly until it runs out of listing, and then it asks if you are finished. That's all there is to the main program folks. And now to the sub-programs that do all the work.

```
260 SUB TITLES
270 CALL CLEAR :: CALL SCREEN(11):: DISPLAY AT(12,6)BEEP :"PRINTER LISTING"
280 SUBEND
290 SUB OPTIONS(S$,P$):: DISPLAY ERASE ALL :: CALL TXTCOL(16,5)
300 CALL FILENAME(1,2,"Edit as needed and ENTER","N?")
310 CALL FILENAME(4,4,"Source file for listing",S$)
320 CALL FILENAME(8,4,"Printer devicename",P$)
330 CALL YN(" Change mind ?","N",22,5,I):: IF NOT(I)THEN CALL HCHAR(22,1,32,64):: GOTO 300
340 DISPLAY ERASE ALL :: IF S$="" OR P$="" THEN DISPLAY AT(1,2)BEEP:"NO INPUT/OUTPUT POSSIBLE" :: CALL DELAY(500) :: GOTO
300
350 OPEN #1:S$,DISPLAY ,INPUT ,VARIABLE 80 :: OPEN #2:P$,DISPLAY ,OUTPUT,VARIABLE 80
360 SUBEND
```

TITLES here is little more than the barest stub, but you can fill that out to your own fancy. OPTIONS takes down the file names, does some checking, and opens the files.

```
370 SUB PAGEBUFFER(PRLN$(,),EFL)
380 REM * New Col Entry *
390 PLN=6 :: COL=COL+1 :: IF COL>2 THEN COL=0 :: SUBEXIT ELSE PRINT "":"** Reading column #";COL:"":""
400 REM * New Para Input *
410 IF EFL THEN PRINT "":" *":"*** END of FILE ***":" *":"" :: SUBEXIT ELSE CALL BASICLINE(NEW$,EFL):: PRINT NEW$:""
420 CALL WRITECOL(PLN,COL,PRLN$(,),NEW$)
430 IF NEW$="END of COL" THEN 390 ELSE 410
440 SUBEND
```

The new column entry in PAGEBUFFER resets the line counter PLN to top of page with a margin, increments the column count, and exits back to the main program if the page is full. If not it tells BASICLINE to fetch a new program line and WRITECOL to enter it in the page buffer. If BASICLINE says it has read the last line it exits and lets the main program worry about that, otherwise it gets another Basic line or starts a new column. A stub here, CALL SKIPLINE(NEW$,SK), could have uses.

```
450 SUB BASICLINE(N$,E)
460 N$="" :: IF NX$="" THEN LINPUT #1:NX$
470 N$=N$&NX$ :: IF LEN(NX$) <80 OR EOF(1)THEN NX$="" :: E=EOF(1):: SUBEXIT ELSE LINPUT #1:NX$
480 PX=POS(NX$," ",1):: IF PX<2 OR PX>6 THEN 470
```

```
490 P=POS(N$," ",1):: IF PX<P THEN 470
500 NR=-1 :: FOR I=1 TO PX-1 :: C=ASC(SEG$(NX$,I,1)):: NR=NR AND C>47 AND C<58 :: NEXT I :: IF NOT(NR)THEN 470
510 IF SEG$(N$,LEN(N$),1)=" " THEN 470
520 IF VAL(SEG$(NX$,1,PX-1))<VAL (SEG$(N$,1,P-1))THEN 470
530 REM ** Check Quotes
540 NQ,I=0
550 I=POS(N$,CHR$(34),I+1):: IF I THEN NQ=NQ+1 :: GOTO 550 ELSE IF NQ<>2*INT(NQ/2)THEN 470
560 SUBEND
```

The procedure BASICLINE which retrieves complete lines of Basic code from the LIST-file is the only part of the program with decision flow complex enough to warrant planning out separately on paper beforehand. I am not going to reproduce this here, but you can work out your own and see if it leads to similar code. The problem comes when the procedure has read in a line exactly 80 characters long. Does the next LIST record then represent a continuation of the same line of Basic or is it the start of a new Basic line? This difficulty can't be ignored if screen list format is to be preserved since 28 into 80 does not go exactly. The procedure provides a cascade of tests each of which checks whether the record being scrutinised should be appended as a continuation of the previous Basic line. A few more rare cases could be tested for along the same lines. There is at least one (that I know of) unlikely case which BASICLINE cannot resolve even in principle. Can you spot it ? It does seem to work well already though. The intricate input code is needed since a VARIABLE file can only be read sequentially, and if the battery of tests says that the last record LINPUTted does start a new Basic line, then this must be saved till BASICLINE is called the next time.

Just be thankful for static variables in XB subprograms ! You also have to take care not to set off the End of File alarm prematurely.

```
570 SUB WRITECOL(P,C,P$(,),N$):: IF NC THEN P=6 :: NC=0
580 IF P>=57 THEN N$="END of COL" :: NC=-1 :: SUBEXIT
590 CALL WRITEPAR(P,C,P$(,),N$)
600 SUBEND
```

Now that WRITECOL has the line of Basic it sends it off to be formed into a paragraph. This simplified program handles coming to the end of a column in a slightly wasteful way that is very simple to program. A normal XB program line lists at most on 5 screen lines, and no matter how tricky you are in entering longer lines the program has already limited it to a string variable (max length 255 or 10 screen lines) or has crashed with an error. The simple minded solution is to exit with End of Col message if the proposed starting line for the new paragraph is past a fixed place somewhat short of the end of the column. The value entered, line #57, is a compromise between making the program totally bulletproof or wasting space. A better approach is to print as far as possible, testing each new paragraph to see if it fits, and if not, holding it over for the next column. If you wondered why the string was called NEW$, then spare a thought for OLD$ which which vanished without trace during program simplification for tutorial purposes.

```
610 SUB WRITEPAR(P,C,P$(,),N$)
620 P=P+1 :: IF LEN(N$)>28 THEN P$(P,C)=SEG$(N$,1,28):: N$=SEG$(N$,29,LEN(N$)-28):: GOTO 620 ELSE P$(P,C)=N$ :: N$=""
630 SUBEND
```

Sub-program WRITEPAR almost was called SALAMI as it slices up NEW$ and assigns the slices to successive printlines. Once entered line 620 loops on itself recursively until the remaining piece fits on a screen line. It assumes range checking has been done before entry. In retrospect, 5 years later, that still looks a fine line of code.

```
640 SUB PRINTPAGE(P$(,),D$)::PRINT "":"** Page print started"
650 PRINT "":"** Assembling printlines":" and printing to" :: PRINT "":"":D$
660 FOR I=1 TO 66 :: PRINT #2:TAB(9);P$(I,1);TAB(45);P$(I,2):: P$(I,1),P$(I,2)="" :: NEXT I
670 SUBEND
```

Not much needs be said about PRINTPAGE beyond noting that line 660 formats a single print record from the two column entries and erases the page buffer as it goes.

```
680 SUB YN(A$,B$,R,C,X)
690  DISPLAY  AT(R,C)BEEP:A$&"(Y/N)"&B$  :: ACCEPT AT(R,C+LEN(A$)+7)VALIDATE ("YN")SIZE(-1)BEEP:A$ :: X=A$=B$ :: R=R+2 ::
SUBEND
```

```
700 SUB KEYCON :: DISPLAY AT(24,6)BEEP:"ANY KEY TO PROCEED"
710 CALL KEY(3,I,ST):: IF ST=0 THEN 710 ELSE DISPLAY ERASE ALL
720 SUBEND

730 SUB FILENAME(R,C,M$,D$)
740 DISPLAY AT(R+1,C):RPT$("-",LEN(M$)):: DISPLAY AT(R,C):M$ :: IF D$<>"N?" THEN DISPLAY AT(R+2,C):D$ ELSE SUBEXIT
750 ACCEPT AT(R+2,C)SIZE(-15)BEEP:D$ :: SUBEND

760 SUB MORE(NM):: DISPLAY ERASE ALL :: CALL TXTCOL(3,12):: CALL YN("More listings","N",16,2,NM):: SUBEND

770 SUB DELAY(A):: FOR A=1 TO A :: NEXT A :: SUBEND

780 SUB TXTCOL(A,B):: CALL SCREEN(B):: FOR I=0 TO 12 :: CALL COLOR(I,A,B):: NEXT I :: SUBEND
```

The FILENAME routine writes an underlined heading, DISPLAYs the default response, and ACCEPTs the reply. If it is asked no question, "N?", it expects no answer. The other SUBs just do their job when called. YN acts like input routines familiar in other TI modules.

```
790 SUB SPEAK(A$):: CALL PEEK(-28672,SP):: IF SP=96 THEN THEN CALL SAY(A$) ELSE CALL DELAY(5*LEN(A$))
800 SUBEND
```

This is a last little goodie tagged on so that you may add speech prompts to your program where desired. A bald CALL SAY has the annoying behaviour that it seems to take forever in giving up the attempt if no speech synthesizer is attached. Line 800 checks that speech is connected and line 820 substitutes a controlled delay if not. CALL SPEAK("....") can then be inserted anywhere it is wanted in the program.

So there we have it, a worked out example of a non-trivial and useful program that makes essential use of the sub-program facility of XB. It shows that the XB programmer can, with a style that finds natural expression in the language without undue contortions, follow the general principles of "structured programming" without getting hung up in the Swiss straight-jacket so beloved by some proponents. The program as presented is a cut-down version of the all-singing, all-dancing model, COLIST, which has now grown to >22K and uses 48 subprograms. In all the versions, subprograms have been an essential tool for program development. Now it's time to take retrospective look at what at what we have done and chase a few more subtleties

RAMBLING BYTES

By "Mac"

Well children, here is another Halloween almost upon us and it seems only a couple months ago I was handing out candy for the last one! My how time does fly. Just another year of TI enjoyment!

You know, thinking back, we use to have a gala time on that night. I came from a small town and it seems everyone would turn out and dress for the occasion. They would attach pillows under the rear of pants and dresses and we boys would always have a paddle. It was so much fun to run up and take a smack at a rear-end with the padding in it and be chased down the street. How much things have changed. Now the big thing is how much destruction can be done with out getting caught by the police. That's not all that's changed either. Some of us can remember when you could buy a bottle of dope for a dime, but it was to brush on a model ariplane and not to destroy your body with. I was watching my 6 year old grandson playing the Wheel Of Fortune on the TI yesterday and it made me think..if my kid brother and I had something like that to play with when we were young, we would have thought we had died and gone to Heaven! If a teen was fortunate enough to have some kind of car then and could go out on a date, it was accepted that he would have a pack of cigarettes in his shirt pocket. Now when a guy starts out in his $20,000 dollar convertable, his mom asks: "Junior, what have you got under your jacket?" And the kid says:"oh, it's only my shoulder holster mom." She says: "oh thank goodness, I thought it was a pack of cigarettes!" But as Bob would say, what has this got to do with computing "Mac"? And he is right.

Thanks to George McCormick, I now have a decent printer. I can now print pictures and have more than two pages of print without the printer putting out garbage for me. George gave me a call a few weeks ago and advised he had an Epson MX/80 he would like to sell. We agreed on a good price and he brought it over the next morning. As you probably know, I had been using the AT&T serial printer which I have been told was a Cito prowriter, but it wasn't compatable to anything I know of! I asn't able to repair the NP-10 Star that Bob gave me and I guess it will never be fixed. The reason being, Star wants $32.50 for the schematic for their printer (most other printer companys give it to you in the user manual) and I just can't see that uch money just for the Tec Manual. I called a local radio supply and asked if they carried a Photofax on it. After a few minutes of checking his list, the clerk called out a number of Star specs he had but no NP-10 of course. Anyhow, the Epson is fine with me. Now the only problem I have is going back through many, many disks that I have changed the default of "PIO" to RS232.BA=4800, and changing it back to PIO!! If you have ever tried to change the default in Funnelweb to PIO, then you can understand just what I am talking about! The DM 1000 is not that rough. Bob told me an easy way to do that....just wait until the main screen comes up and hit "F/3" and it will step you right through the process. You may even send print codes and all of it can be saved to disk.

After not hearing from the K-Town group since early spring, we got the October issue of their newsletter and there was a very interesting listing on the back page. It consisted of 1-800 numbers for BBS's all over the country. I noticed two from N.C. and they both were in Raleigh. Unfortunately, the list was copied from the November 1988 issue of Computer Shopper and it was in sub- script so it was very hard to make out the numbers. If any of you are fortunate enough to have the November 1988 issue, I would appreciate knowing about it. I would like to have a copy I could read because I would like to call some of the boards and see if they cater to TI. I will list a few here that I can make out but I still might have one of the numbers wrong. I am using my OPTIVISOR on them and that makes the print a little larger. Anyhow, I do thank the K-Towners for giving out the info.

Here are a few of the 1-800 numbers

California:
Sunnyvale 1-800-533-3177
San Diego 1-800-854-2003
Los Angeles 1-800-421-4000

Colorado:
Denver 1-800-525-7877
Denver 1-800-663-0940

Delaware:
Wilmington 1-800-441-7680

Florida:
Ft. Laud. 1-800-327-6033
Pinelas Park 1-800-237-0010
Orlando 1-800-327-0204

Kansas:
Wichita 1-800-835-1120

Michigan:
Pontiac 1-800-392-6881
Ann Arbor 1-800-393-3705

North Carolina:
Chapel Hill 1-800-334-SOFT
Chapel Hill 1-800-334-5470

New York:
Spring Valley 1-800-431-2816
Rochester 1-800-828-6772

New Jersey:
West Milford 1-800-520-5313

Oklahoma:
Bethany 1-800-654-4058

Maine:
Burlington 1-800-343-0873

Texas:
Dallas 1-800-824-7555
Houston 1-800-231-6671

These are just a few of the many listed. Like I said, the print was very small so if I might have gotten a number wrong, forgive me. At least I tried! (hint,hint)!

It was unfortunate indeed to hear of the untimely demise of John Guion. Although I have never had the pleasure of meeting him, I am familiar with some of his gifts to the TI community. It is a loss to us and I send our condolences to his family. We also lost Guy Romano who ran the Amnion Helpline. There is no telling how many users he has helped. We will eatly miss these two "brothers" in years to come. We also got the bad news that Scott Copeland of the EAR in Great Britian as been run-down by a car and is having a pretty rough time of it. According to Joanne, they will be a little late with the newsletter for a while. Here's wishing you a speedy recovery Scott.

Well, Bob is in a hurry to get this so, until next month, enjoy the good Times and see you at the November meeting.

PS...watch those witches & goblins!

FOR SALE

Buddy Cato has the following for sale:

1 Console
1 P.E. Box w/32K, RS232, Single Sided Disk Drive and Controller
1 Speech Synthesizer
1 Gram Kracker Extended Version
1 Extended Basic Cart.
The following cartridges:
Editor Assembler, TI/Writer, Personal Record Keeping, Multiplan, DMII, TI Logo, Terminal Emulator II, Household Budgeting and books on TI.

Buddy is asking $550.00 for the entire lot and he does not wish to break up the components. He is open for offers and may be reached at: (919) 288-5054.

TONY'S CORNER

Confessions of a Small-Time User. by Tony Kleen. Guilford (NC) Users Group.

Let's learn a little more TIBASE with examples. I've been writing some programs for our neighborhood club. We've got monthly dues, collections, dunning; the whole gamut. Got the statements out earlier, now I want to see how we're doing on member's balances at the end of the month. This is the programs that I'll be explaining this month.

First, let me tell you about the ACTIVITY database. The fields contained on the db are as follows: (O)wner, (M)em(B)e(R), (TRANS)action (DATE), (TRANS)action (DESC)ription, and (TRANS)action (AM)ou(NT). I have one BLANK character at the end of the record. The other db being used is one called MBR_HIST, short for member history. I know, you had already figured that one out, right?!? The main fields here are the twelve monthly 'buckets' for holding numeric values. The index (SORT ON) field is expected to be the field called KEY. This field is made up of several sub-fields, mainly the Owner, the MemBeR, the YEAR, and a constant value "MTHEND". I plan to have other member history records on this db, not just their MTHEND values. To add other history records, all I have to do is come up with a different suffix in my KEY.

Now, let's get into the program B5! At the end of each month, I'm interested to know what each member's balance is, as well as the membership as a whole. I've set up the history db mainly for reporting purposes. One can do trend analysis for any period of time, for any member, or for the club as a whole.

You'll notice that the program is separated into four segments; locals, select, process, and close. Within the local segment, I define my variables in a specific way, the 1st character of the variable tells me the type, the suffix tells me the number . 'C13' for thirteen characters. N8 for eight numerics. Now, what's that strange K3 animal. K is for constant. 'C' was already used for character!
The second segment, select, is where I define and open the databases. As you can see, slot 2 is for the ACTIVITY, slot 3 is for MBR_HIST. The last segment simply closes the databases.

The process segment is where the 'work' gets done. As you might notice, I DO (or execute, or perform) three command files; B51, B52, B53. Each of these command files is a WHILE loop, and we're in these loops for much of the program's duration. Be advised that there is overhead to DO'ing a command file. The system has to exit the command file it's currently in, set a line pointer, then load the CALL'ed command file. If you call a command file for only one pass through it, you would be time ahead to incorporate the call'ed command file into you call'ing command file.

Back to the process. The first directive is to turn the database's SORT OFF. This will require a pass through the database to reset the index! It's a short duration, though, and well worth the time in accessing the records sequentially, later in B51. Immediately after the sort, one needs to position the database at the TOP record. The sort just leaves you at EOF! In B51, you notice that WHILE we do NOT have an EOF condition, we REPLACE our AMT with zero, and MOVE one record forward. In otherwords, we're zeroing the work field called AMT! Our next command file will be to accumulate the member's activity to arrive at his month-end balance. B52 DOes this function. Again, WHILE the ACTIVITY db is NOT at an EOF condition, we cumulate the TRANSAMNT into the AMT field. The 'IF 2.D = "B"' needs some explanation. Simply, the Owner is 'B'rookcliff, he name of the club. The TRANSDATE is being compared to the current DATE, looking for any date less than, or equal. The xt REPLACE/SELECT/FIND combination builds our index, then searches the proper db for our record. If we don't find the ecord, we build one. We then, of course, accumulate our AMT and go get the next ACTIVITY record.

You have probably wondered why I'm using a work field on the MBR_HIST db to accumulate the month-end balance, especially when I've got a field allocated for that balance. The command file B53 can explain that. After I have accumulated all the ACTIVITY, I pass through the MBR_HIST db and move the AMT value to the proper JAN,FEB,etc field. By having the work field, I need only pass through the DOCASE/ENDCASE construct once. With 200 members, I execute the 'REPLACE AMT WITH' statement 200 times, instead of the 38 statements in the DOCASE construct 200 times. This saves lots of time!

Notes to remember. After SORT'ing a db, reset to TOP.
WHILE (.NOT.(EOF))/ENDWHILE constructs should be considered for their own command file.
Standardize your LOCALS; (C), (N), (K), etc.
When creating a database, add BLANK characters to the end, and design one unique index field.

```
* BROOKCLIFF.B5/C
   10/04/89
* --------------------------
------------
* MBR Month-end balances.  (
.DATE.)
* --------------------------
------------
*
 L O C A L S
LOCAL C13 C 13
LOCAL K3  C 03
REPLACE K3 WITH (YEAR(.DATE.
))
* --------------------------
------------
*
 S E L E C T
SELECT 2
USE ACTIVITY
SELECT 3
USE MBR_HIST
* --------------------------
------------
*                          P
 R O C E S S
SORT OFF
TOP
DO B51
SORT ON KEY
*
SELECT 2
SORT OFF
TOP
DO B52
*
SELECT 3
TOP
DO B53
* --------------------------
------------
CLOSE ALL
RETURN
*
  C L O S E
* --------------------------
------------
* --------------------------
------------
* BROOKCLIFF.B51/C
   10/04/89
* --------------------------
------------
WHILE (.NOT.(EOF))
  REPLACE AMT WITH 0
  MOVE
  ENDWHILE
RETURN
* --------------------------
------------
* --------------------------
------------
* BROOKCLIFF.B52/C
   10/04/89
* --------------------------
------------
WHILE (.NOT.(EOF))
  IF 2.D = "B"
    IF (.NOT.(TRANSDATE > (.
DATE.)))
      REPLACE C13 WITH D:MBR
!K?
      SELECT 3
      FIND C13
      IF (EOF)
        APPEND BLANK
        REPLACE   KEY WITH
 C13
        REPLACE 3.MBR WITH 2
.MBR
        ENDIF
      REPLACE AMT WITH AMT +
 2.TRANSAMNT
      SELECT 2
      ENDIF
    ENDIF
  MOVE
  ENDWHILE
RETURN
* --------------------------
------------
* --------------------------
------------
* BROOKCLIFF.B53/C
   10/04/89
* --------------------------
------------
WHILE (.NOT.(EOF))
  DOCASE
    CASE (MONTH(.DATE.)) = 1
      REPLACE JAN WITH AMT
      BREAK
    CASE (MONTH(.DATE.)) = 2
      REPLACE FEB WITH AMT
      BREAK
    CASE (MONTH(.DATE.)) = 3
      REPLACE MAR WITH AMT
      BREAK
    CASE (MONTH(.DATE.)) = 4
      REPLACE APR WITH AMT
      BREAK
    CASE (MONTH(.DATE.)) = 5
      REPLACE MAY WITH AMT
      BREAK
    CASE (MONTH(.DATE.)) = 6
      REPLACE JUN WITH AMT
      BREAK
    CASE (MONTH(.DATE.)) = 7
      REPLACE JUL WITH AMT
      BREAK
    CASE (MONTH(.DATE.)) = 8
      REPLACE AUG WITH AMT
      BREAK
    CASE (MONTH(.DATE.)) = 9
      REPLACE SEP WITH AMT
      BREAK
    CASE (MONTH(.DATE.)) = 1
0
      REPLACE OCT WITH AMT
      BREAK
    CASE (MONTH(.DATE.)) = 1
1
      REPLACE NOV WITH AMT
      BREAK
    CASE (MONTH(.DATE.)) = 1
2
      REPLACE DEC WITH AMT
      BREAK
    ENDCASE
  MOVE
  ENDWHILE
RETURN
* --------------------------
------------
                 MBR_HIST     /
00056
   CREATED 09/30/89 CHANGED
09/30/89
  FIELD  DESCRIPTOR  TYPE  W
IDTH  DEC

     1   KEY           C
013
     2   MBR           C
003
     3   AMT           N
008    02
     4   JAN           N
008    02
     5   FEB           N
008    02
     6   MAR           N
008    02
     7   APR           N
008    02
     8   MAY           N
008    02
     9   JUN           N
008    02
    10   JUL           N
008    02
    11   AUG           N
008    02
    12   SEP           N
008    02
    13   OCT           N
008    02
    14   NOV           N
008    02
    15   DEC           N
008    02
    16   BLANK         C
013

* --------------------------
------------?
                 ACTIVITY     /
00335
   CREATED 89/09/16 CHANGED
10/01/89
  FIELD  DESCRIPTOR  TYPE  W
IDTH  DEC

     1   D             C
001
     2   MBR           C
003
     3   TRANSDATE     D
008
     4   TRANSDESC     C
036
     5   TRANSAMNT     N
008    02
     6   BLANK         C
001
^^^^^^^^^^^^^^^^^^^^^
```