THE GUILFORD 99'ER NEWSLETTER

VOL.4 NO.12 DECEMBER 1987

Mack Jones, President
Mike Garrett, Secretary/Treasurer
BBS: (919)274-5760(OPUS)

Robert Carmany, Vice President
Robert Dobo, Program Library
ROS (919)-621-2623

The Guilford 99'er Users' Group Newsletter is free to dues paying members (One copy per family, please). Dues are $12.00 per family, per year. Send check to 3202 Canterbury Dr., Greensboro, NC 27408. The Software Library is for dues paying members only. (Herman Geschwind, Editor)

OUR NEXT MEETING

DATE: December 1, 1987. TIME: 7:30 PM PLACE: Glenwood Recreation Center 2010 S. Chapman Street.

December is the time for our annual traditional get-to-gether with appropriate music (computer generated), good food (which we expect you to contribute), companionship (be sure to bring your spouse, girl friend, boy friend) and exchange of gifts (more specifically we will continue our swap fest from last month with new disks to copy). Last and not least we will also elect new officers for the next year. Be sure to attend!

PRES PEEKS

The time of the year is here when you don't really know from one day to the other what the weather will bring, but something must have been right for the last meeting! In spite of the polls being used in our regular meeting room and a basket ball managers meeting in our alternate room, we filled the kitchen full and overflowing! I sure was glad to see so many members back. Just couldn't believe my eyes when Buddy and Wife strode in! Welcome back Buddy.

While we're on the subject of welcomes, I would like to welcome our newest member Mr. Walter Tietjen Jr. from down Raleigh way. Welcome Walt, and we look forward to serving you. Welcome to Mr. William H. Woodruff from Burlington. Bill is a former member of Winston Salem and has recently moved to Burlington. We are fortunate to lie between Burlington and Winston, so we get Bill!! Also, we have a renewal from Mrs. Ellen Kramer from up New Jersey way. Welcome back for another year Ellen. Let's welcome these folks all you members who have not already done so.

I am looking forward to the Christmas meeting next month. Every December meeting, we have loads of goodies, thanks to the unsung members wives. Even tho they (most of them) are never seen at the meetings, they help out when needed. I encourage each member to bring their Spouse for the December meeting so we can all get to meet the "other half". I also encourage members who have written any Christmas programs whether it be music or graphics or both, to bring them along. There will be a PEB with drives available so we can just have a rooting-tooting good time of fellowship.

At the December meeting, Herman will have the nominations for our new officers, so please try to attend and cast your vote. I also urge you, if you are nominated, have a go at the office you are nominated for. It is an honor to represent your

club, and you will get a lot out of it, I know. Thanks. I know that I have enjoyed serving you as President and hopefully, can serve you in another office in the future. After all, it's your club so you should have a chance to be a part of it's running.

Hopefully, all members will be able to join us on the 1st of December, but for those of you who simply can't come, let me be one of the first to wish you a very, very, Merry Christmas, and may 'ol Santa fill your stocking with good TI goodies!! 'Yall come. (Submitted by L.F. "Mack" Jones, Pres.)

ONLY YESTERDAY - -

Housing cleaning recently, I ran across some old issues of Home Furnishings Daily and its special section called ComputerWare, which I used to receive in connection with the furniture market. In one, February, 1983, I found a most interesting article on Bill Turner, president of Texas Instruments consumer products division in 1982.

Thought ya'll might enjoy some choice excerpts, like one unidentified industry analyst's suggestion that this division had earned TI's "major, long-term commitment, still critically dependent on Turner himself." Remember this was written in Spring, 1983, and the article credits Turner as being the "marketing sharpshooter" behind TI's first $100 rebate announcement, bringing the 99/4A down to $199 on Aug. 4, 1982. The article's lead paragraph calls it a "red-letter date," and I think we could all agree that red is quite the right adjective.

The article notes that Turner exceeded his goal of having the 99/4A on 7,000 outlet shelves by the end of 1982. The figure was closer to 10,000, and his "10-gallon goal," according to the article, for the infamous 1983 was to have the computer available through 25,000 outlets. So much for volume as the critical factor in mass-marketing.

To the "storm of criticism about profit-busting implications of the rebate," Turner told the trade journal, "It's just a matter of time before we see the traditional retailer sell professional and small business computers," and "Next year everyone will broaden their line to include several (price) points that will sweep upward." This was at a time when TI also was introducing its 99/2, priced at $99 to compete with the Timex, et. al.

Turner also talked at length about "feature-rich" consoles with more options; software as the critical component; multiple positions of use (student, hobbyist, entertainer, educator, at home professional); a more educated consumer "who has gone beyond price point and kilobytes as the only criteria for purchase; new distribution channels, like drug and video stores; professional computers "creeping" out of computer stores into traditional retail outlets and (get this) "a growing aftermarket--wide open territory for those willing to pioneer it."

Prophetic, no?

How about these golden nuggets: TI is "in the computer business for the duration;" "No matter what skirmishes may develop among the retailers in 1983, Turner and TI are poised for the best year yet;" "The competition this year will be different than last year's--we'll be competing on feature, not price;" and "the success of the 99/4A home computer is being viewed as a prototype for a turnaround for the entire (TI) company." Remember now, these are the pros talking, the guys with MBAs and analyst credentials.

The article, which I will try to have at the next meeting for anyone wishing to read it completely, also has a fleeting reference to what I guess was the 99/8 successor "to compete with the Commodore 64 and the Atari 1200."

Personally, I really like the article's ending line: "The only thing certain is that Bill Turner will be doing his best to stay on top of his bucking bronco of a market." Wonder where he landed?

Here's a couple of other kernels I culled from that issue:

--according to a national distributor's "hit list," the top five TI-99/4A software offerings in February 1983 in four categories were: Munchman, Parsec, TI Invaders, Car Wars, and Tombstone City in entertainment; Addition and Subtraction I, Early Learning Fun, Number Magic, Addition and Subtraction II and Multiplication I in education; Household Budget, Home Financial Decisions, Personal Record Keeping, Personal Real Estate, and Tax/Investment Record Keeping in home/business; and Teach Yourself Basic, Extended Basic, Terminal Emulator II, Mini-Memory and Editor/Assembler in utility. It is interesting to note that all are TI cartridges, which contrasted significantly with lists for other computers that had a range of third party

are citations. Wonder what such a list today would contain!

--Computer Research Analyst Alex Stein, commenting on the TI and Commodore console price cuts, said, "Today's home user is not sophisticated enough to appreciate the differences between systems. He does know that 64K versus 16K let's you use more professional applications, but he's still buying a console, not a system."

--An unidentified retailer observed about the difficulties of merchandising home computers, "The only thing that makes it all worthwhile is that consumer demand is so great. The sales projections in general are just incredible, and I believe they will prove accurate. In a way, that gives all of us a lot of room for mistakes."

--And on a lighter note, a reference to a New York Times report on "computer divorce." That article, titled "New Marital Stress: The Computer Complex," reported that "Families suffer when a spouse is obsessed with the perfect machine, but still dealing with the imperfect human being." Worse the machine is charged with creating "computer widows" who are becoming commonplace "in psychotherapists' offices complaining of neglect and maintaining their mates are mesmerized by the machine." The article asks whether the machine might someday be cited as a "co-respondent" in a divorce case.

Can charges of alienation of affections be far away? Shouldn't we consider a cooperative defense: having a club seminar on programming these machines to testify on our behalf, say about our exemplary characters, for example. Surely the court will take into consideration our humanity in adopting the abandoned 99/4A. And what about TI's joint responsibility--not to mention Bill Cosby's--in getting us addicted in the first place. It's getting serious, don't ya think. (Contributed by Larry Spohn)

FORTH FORUM

Another deadline is here and I haven't even started to write my monthly column. I guess I'll have to see what I can find that might be of interest to you all. If memory serves me, last month we had something in TI-Forth. That means that it is time to try a Mycove Forth application. How about a few screens that could be used as the basis for an RS232 terminal program? Well, let's start at the beginning!

These screens were among a series that I got "way back when . ." from Tim MacEachern the developer of Mycove Forth. They were released to anyone who bothered to write and ask for some Forth material. Without any further delay, here they are ---- with comments. The only pre-requisite is that Screen #11 on the distribution disk which loads PAB: be loaded before this application is used.

```
0 ( Write a character to an RS232 port )
1
2 : C!RS232 ( c port # -- : write char
3           to RS232 port 1 or 2 )
4 ( Convert the port # to a CRU address )
5 32 * >980 + ( 9902 CRU bit address )
6 ( Turn on request to send - RTS )
7 1 OVER 16 + 1 !CRU
8 ( Wait until DSR and XBRE are set )
9 BEGIN
10 ( Test Data Set Ready )
11 DUP 27 + 1 @CRU
12 ( Test Transmit Buffer Reg Empty )
13 OVER 22 + 1 @CRU
14 ( Wait until both signals are high )
15 AND UNTIL
16 ( Write the character to the Trans buffer )
17 SWAP OVER 8 !CRU
18 ( Reset Request to Send )
19 0 SWAP 16 + 1 !CRU
20
21 ( Example : 65 2 C!RS232 sends an 'A'
22 to device RS232/2. Use OPEN to set
23 transmission attributes. ) -->
```

```
0 ( Read a character from an RS232 port )
1 ( Check the port.  If no character is
2 there then return a 0 flag.  Otherwise read
3 the character and return a 1 flag.  )
4 : ?C@RS232 ( port # -- c 1 : got a char
5                or -- 0 : no char )
6 ( Compute the RS232 CRU base address )
7 32 * >980 + >R ( 9902 CRU base addr )
8 ( Test for Data Read Ready )
9 R 27 + 1 @CRU
10 ( And in Receive Buffer Loaded )
11 R 21 + 1 @CRU AND
12 ( Read the char, if one is there )
13 DUP IF
14 ( Read the char, mask off parity bit )
15 R 8 @CRU >7F AND
16 ( Reset the Receive Buffer Register )
17 0 R 18 + 1 !CRU SWAP
18 ( Return, dropping the stored CRU base )
19 ENDIF R> DROP ;
20 21 ( Example: to check RS232/2 for a char
22 2 ?C@RS232 will return either 0 or
23 the char and a 1 flag.  ) -->
24
```

```
0 ( ?KEY - check keyboard for a keypress )
1
2 ( This code allows for automatic
3 repeating of keys held down.  )
4       20 CONSTANT K-REPEAT
5      180 CONSTANT K-INIT
6 K-INIT VARIABLE K-COUNT
7
8 : ?KEY ( -- c 1 : if a key is pressed
9       or -- 0 : if no key is pressed )
10 ?KEYBOARD CASE
11 ( No key pressed )
12     0 OF K/INIT K/COUNT ! DROP 0 ENDOF
13 ( New key pressed )
14     1 OF K-INIT K-COUNT ! 1 ENDOF
15 ( Same key, still pressed )
16     K-COUNT +!
17 K-COUNT @
18     IF DROP 0
19     ELSE K-REPEAT K-COUNT ! 1 ENDIF
20   0 ENDCASE ; -->
21
22
23
24
```

```
0 ( A simple terminal program )
1
2 ( File to set transmission protocol )
3 1 CONSTANT RPORT# 11 -LOAD PAB:
4 1 16 PAB: RPORT RS232/1.PA=N.DA=8
```

```
5 ( Exit character is CLEAR - FCTN-4 )
6 2 CONSTANT EXIT-CHAR
7 : TERM-A ( -- : act as a terminal on
8             RS232 port #1.  Pressing
9             EXIT-CHAR exits terminal.)
10 RPORT OPEN ( sets protocol )
11 BEGIN
12 RPORT# ?C@RS232 ( test the port )
13     IF ( print the char if not NULL )
14       -DUP IF EMIT ENDIF
15     ENDIF
16     ?KEY ( test the keyboard )
17     IF ( test for exit-char pressed )
18       DUP EXIT-CHAR = IF QUIT ENDIF
19       RPORT# C!RS232 ( write to port )
20     ENDIF
21   AGAIN ;
22
23
24
```

BASIC CORNER

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 DISPLAY/VARIABLE 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 will do only the minimum necessary to do the job properly. Bells and whistles can be added later. In one or two places we shall 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, 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. 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.

```
(13)
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:N
EW$
200 REM * NEW PAGE ENTRY *
210 CALL PAGEBUFFER(PRLN$(,),ENDFILE)
220 CALL PRINTPAGE(PRLN$(,),PDEV$):: IF ENDFILE=0 THEN 2
10
230 REM * END OR NEXT *
240 CLOSE #1 :: CLOSE #2 :: CALL MORE(NM):: IF NM THEN C
ALL 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

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 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 TXTCO
L(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$(NX$,1,1)):: N
R=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 5
50 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 drawing out a flow diagram beforehand. I am not going to reproduce this he 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 lin 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 lines of 540-550. There is 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.

```
640 SUB PRINTPAGE(P$(,),D$):: PRINT "":"## Page print st
arted"
650 PRINT "":"## Assembling printlines";" and printin
g 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 PROCE
ED"
710 CALL KEY(3,I,ST):: IF ST=0 THEN 710 ELSE DISPLAY ERA
SE 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 SUBEX
IT
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$):: CALL PEEK(-28672,SP):: IF SP=96 TH
EN THEN CALL SAY(A$) 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

MODEM TALK

The most exciting news this month is that two of our bright gurus, Al Beard and Barry Boone have come up with a new version of ARChiver that will save us time and money. The archivers that were available up till now were a great help since they combined any number of related files into one package for downloading. Al and Barry, independently of each other came up with the idea to take this one step further and to compress the file so that there is less to download. As an example, the #2 Funnelweb disk has 334 sectors. Just archiving all the files cuts this down to 318 sectors (just by eliminating individual file headers). Al's compression system cuts this down to 243 sectors and Barry's system to only 197. In terms of efficiency, speed and ease of use, Barry Boone's Archiver II 2.3 is clearly the winner. From now on we will use this system for all of our uploads. The file will have the code BRC (for Boone's aRCchiver) in the file name to show what system was used.