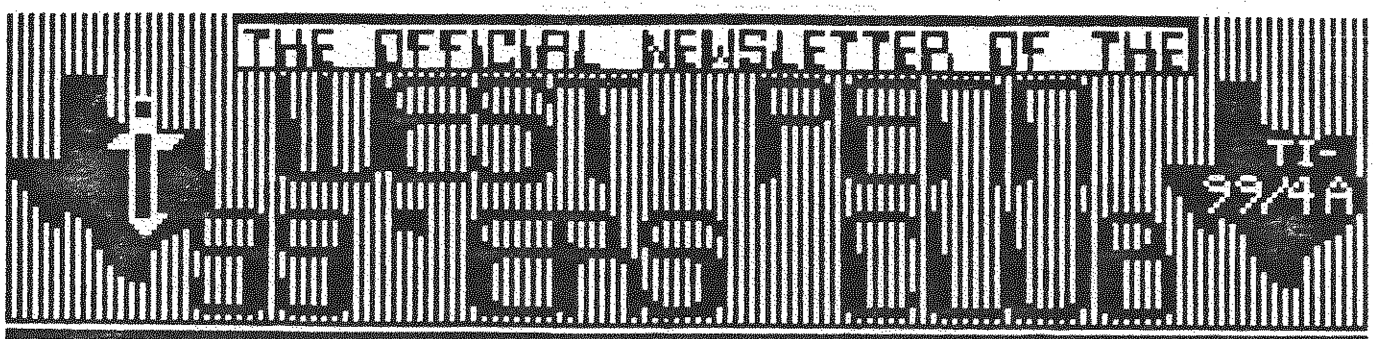


MAY 1986 NO. 8

MY HOW TIME FLIES!
by John Willforth

I've never seen time fly as fast as it does now that I've taken on the responsibility of putting out a newsletter. I used to keep track of various parts of the month by what bill was due. Now the bills and their getting paid , have become but a blurr. I spent my early life on a farm, and by some standards this would cause many people to think that, for me, life was slower than that of my city cousins. Believe me when the dark rain clouds came and 300 bales of hay were still out in the field, there wasn't any city cousin gonna keep up to me! Years and pounds have come along now, and maybe I'd let the hay lay now rather than die of a heart attack trying to get it in.

Gathering information for a newsletter, trying to put it together into some semblance of order that will be new and interesting to the most people possible is what it's all about. I really enjoy it (even if I do hear about it from my family) a lot. If there is anyone out there who wants to contribute to a club, and thinks that they can't write programs, can't design or build hardware, can't teach a S.I.G., you CAN edit a newsletter! There are many user's groups that are struggling only because there is no one to get the word out to the members on what is going on in the TI COMMUNITY, both in your own group and nationally as well as internationally.

You say that you just can't give the necessary time to editing a newsletter, well you could write an article, one that deals with something that interests you, remember, something that you learned is something that others would like to know. I believe that we here in the WEST PENN 99ERS have a very good combination of teachers and preachers to do the job.

I want to say " HI, TOM and BARB HARE out there in TEXAS, miss you back here. Tom and Barb moved to browner pastures. I say that, because you can't have greener pastures than here in Pennsylvania.

I want to apologize to all of you who have gone to have your eyes checked, because you couldn't read the newsletter the last couple of months. This nonsence will stop with this issue. I even had some difficulty, and I wrote it.

We really had a good meeting in April, and because our secretary wasn't there, we don't have an accurate record of it to report to you . Why don't some of you ask him where he was the next time you see him. Hint: Would you believe that a grown man given a stack of newsletters from various TI clubs from around the North American continent and Hawaii, could get so engrossed that he loses track of the TIME, let alone the DAY?

That reminds me we are going to put the newsletters we receive, into a binder and LOAN them out. The penalty for not returning the LOANED out book will be a visit from me. You can take that however you like.

Ed Bittner and I are planning to make a VIDEO on how to put 32K inside your console or speech synthesizer. I know that from the newsletters I've received that many clubs do have people who are doing this, but I'm also finding that in some clubs, there is no one to do this for members. I don't like to beat a dead horse, but I think that the horse is a very long way from dead.

NEXT MEETING:...MAY 19, 1986
NORWIN YMCA
ACHTUNG! 7:00 P.M.

MEMBERSHIP.. $15 PER YEAR FOR INDIVIDULE/FAMILY OR $10 FOR ASSOCIATE (NEWSLETTER ONLY). CALL ANY OFFICER ------->

PRES.......	SCOTT COLEMAN	(412) 271-6283
V. PRES....	CHUCK STRINK........	(412) 668-2811
SECRETARY..	ED BITTNER..........	(412) 864-4924
COR. SECT..	GENE KELLY..........	(412) 829-0469
TREASURER..	JAN TRAYERS.........	(412) 863-1575
LIBRARIAN..	CLYDE COLLEDGE......	(412) 828-3042
EDITOR.....	JOHN WILLFORTH......	(412) 527-6656

TREASURERS' REPORT

BALANCE IN ACCOUNT FOR APRIL 1,1986	$459.52
PAID FOR DISKS,CASSETTES, AND DATA CASES	175.19
RENT TO YMCA FOR MEETING	10.00
RECEIVED FOR SALE OF DISKS, AND DATA CASES	191.50
RECEIVED FOR RAFFLE AT MEETING	23.00
RECEIVED FOR NEW MEMBERSHIP	10.00
PAYMENT FOR ADDITIONAL DISKS AND LABELS	204.92
RECEIVED FROM JOHN FOR SALE OF DISKS	160.00
BALANCE IN ACCOUNT FOR MAY 5,1986	$453.91

BASIC BASICS

By Charles Strink

The TAB function lets you specify a column location where the next piece of data in a PRINT statement will be printed.

Even tho you can tell the computer in which column to start printing, you cannot print past the end of your screen, past the end of a record, or off the edge of the printer. If you use a number of less than 1, the computer will replace it with a 1. If your number is past the end of the line the computer will subtract the maximum length of the lines until it gets a number that it can use correctly.

EXAMPLE:

You are printing on a TV screen and you use TAB(45);. Since a TV screen print is 28 characters long, the computer will subtract 28 from 45 and begin printing in column 17.

If you are already past the indicated column position when the TAB function is encountered, the computer will move to column on the next line.

The TAB function is considered a print item and must be enclosed in print separators in the same way as any other print item. Any print separator befor the TAB is acted upon befor the TAB column is evaluated.

Data items will not be split between screen lines unless the data is a string of more than 28 characters. If TAB is used and your data item is too long to fit onto the line, the data will be printed on the next line, regardless of what you said using TAB.

Example using TAB;

```
10 CALL CLEAR
20 FOR J=1 TO 10 STEP 3
30 PRINT TAB(5);J;TAB(10);J+1;TAB(15);J+2
40 NEXT J
50 STOP
```

Until next time.................................

.................................Happy Computing

THINGS THAT HAVE CROSSED MY MIND......

We still have a lot of Double Sided Double Density diskettes(floppys) left. If you want some, please call JAN TRAYERS and we will see that you get them. Price: 10/$8.

Just because summers coming, don't start to think about replacing the THIRD MONDAY of the month, with fun things, like wash'n the car, or mowi'n the grass.

MYARC is about to almost announce, a nearly ready, nearly fully compatible TI similiar machine. That's about as clear as it is in my mind. But seriously(haven't I been), the new machine will NOT be a machine, but a card that fits in the PEB! Whew, am I glad I stayed with the PEB! The machine will come with half a MEG of memory, and can support more in PEB slots. No cassette support, and no cartridge port. You will be able to dump the cartridges you have to disk and load them this way however. The keyboard will be an IBM type that plugs into the card and the machine will only support 80 colmn. RGB monitors(no TV). Let's wait to see all the TRUTH. It's still interesting to DREAM!

PARAMETER PASSING BETWEEN EXTENDED BASIC PROGRAMS

by Mike Kabala

One of the more advanced features available to the Extended BASIC programmer is the ability to chain programs by using the RUN statement within a program. This allows the creation of software that would not fit into the memory all at once. When you're done with one program segment, just "RUN" the next one.

Unfortunately, there is one bug. The RUN statement clears out all variables even if it is executed from within a program. Type in the following 2 programs and you will see for yourself. Be sure to save the first as "DSK1.DEMO1" and the second as "DSK1.DEMO1A" and then load and run "DSK1.DEMO1".

```
100 CALL CLEAR
110 DISPLAY AT(1,1):
    "TYPE SOMETHING."
120 ! ACCEPT AT statement is
130 ! needed to put text in
140 ! a predictable screen
150 ! location
160 ACCEPT AT(2,1):A$
170 ! All variables will be
180 ! cleared when the next
190 ! statement is executed
200 RUN "DSK1.DEMO1A"
```

```
100 CALL CLEAR
110 ! Variable has been
120 ! cleared by RUN
130 ! statement
140 PRINT A$
```

See what I mean? Somewhere between DEMO1 and DEMO1A, the value of A$ got lost. Now you could, if you wanted to, create a file, store A$ in it, and then read A$ from that file after entering DEMO1A, but that seems rather wasteful if you only need to pass one or two variables. In the remainder of this article, I will show you two other methods I have found to get around the problem.

The first method will work even if all you have is a console and cassette recorder (although you'll have to change all of the program names to "CS1"). That method is to use the screen as auxiliary memory. Just display the information you want before executing the RUN statement and your chained program will be able to take the data right off the screen as demonstrated below.

Load DEMO1 back into your computer and modify its RUN statement to chain in DEMO2A as shown below. Then save the first program as "DSK1.DEMO2" and the second one as "DSK1.DEMO2A". Finally, load and run "DSK1.DEMO2" and what you type in the first program should be correctly printed by the second.

```
100 CALL CLEAR
110 DISPLAY AT(1,1):
    "TYPE SOMETHING."
120 ! ACCEPT AT statement is
130 ! needed to put text in
140 ! a predictable screen
150 ! location
160 ACCEPT AT(2,1):A$
170 ! All variables will be
180 ! cleared when the next
190 ! statement is executed
200 RUN "DSK1.DEMO2A"
```

```
100 ! Read what last program
110 ! left on the screen.
120 ! Offet of 2 is needed
130 ! for difference between
140 ! ACCEPT and GCHAR
150 ! statements
160 A$=""
170 FOR I=3 TO 30
180 CALL GCHAR(2,I,A)
190 A$=A$&CHR$(A)
200 NEXT I
210 ! Parameter has been
220 ! passed using the
230 ! screen as auxiliary
240 ! memory
250 CALL CLEAR
260 DISPLAY AT(5,1):"YOU TYPED:"
270 DISPLAY AT(6,1):A$
```

For this second method you will need to have the 32K memory expansion installed. The trick here is to use the space reserved for an assembly language program for temporary storage. This has the advantage that you don't need to display something to pass it to the next program. Be sure to use the memory betwen 9984 and 16184 after executing CALL INIT because the rest of memory is already in use by your program and other things.

Save the first program as "DSK1.DEMO3" and the second as "DSK1.DEMO3A". Then run "DSK1.DEMO3".

```
100 CALL CLEAR
110 PRINT "TYPE SOMETHING:"
120 ACCEPT A$
130 !
140 ! Reserve space for
150 ! assembly language
160 ! program
170 !
180 CALL INIT
190 !
200 ! Store length of string
210 !
220 A=LEN(A$)
230 CALL LOAD(9984,A)
240 !
250 ! Store string
260 !
270 FOR I=1 TO A
280 CALL LOAD(I+9984,
    ASC(SEG$(A$,I,1)))
290 NEXT I
300 CALL CLEAR
310 RUN "DSK1.DEMO3A"
```

```
100 CALL CLEAR
110 PRINT "YOU TYPED:"
120 !
130 ! Get string length
140 !
150 CALL PEEK(9984,L)
160 !
170 ! Get string
180 !
190 X$=""
200 FOR I=1 TO L
210 CALL PEEK(I+9984,X)
220 X$=X$&CHR$(X)
230 NEXT I
240 !
250 ! Print string
260 !
270 PRINT X$
```

You may have noticed that I have only used string type variables in these examples. That's because the data manipulations required are a bit simpler for string variables. That doesn't mean that you can't pass numeric variables, however. Just use STR$() to convert them to string variables and VAL() to convert them back. I'm sure it wouldn't take too much effort to figure out a way to pass them directly, either. Just be careful if you use the second method that you don't try to pass a number greater than 127 or less than -128 unless you split it up into more than one byte.

Finally, if you don't have a disk drive and want to try the first method, type in the program below and save it to CS1. Then type in the DEMO2A program and save it to the same cassette WITHOUT rewinding it. Then rewind the cassette, load the first program, and run it.

After the program begins running, it will ask you to rewind the cassette. Ignore this message and just proceed to load in the second program or you will end up chaining the first program back into memory.

The reason lines 110 and 160 of the first program had to be modified is that the computer prints 14 lines to the screen when chaining the second program. If this offset had not been accounted for, the second program would have read the wrong portion of the screen.

```
100 CALL CLEAR
110 DISPLAY AT(15,1):
    "TYPE SOMETHING."
120 ! ACCEPT AT statement is
130 ! needed to put text in
140 ! a predictable screen
150 ! location
160 ACCEPT AT(16,1):A$
170 ! All variables will be
180 ! cleared when the next
190 ! statement is executed
200 RUN "CS1"
```

THESE NEXT THREE ARTICLES ARE FOR THOSE WHO ARE USING A MODEM AND ARE LOOKING FOR SOME HELP IN TAILORING THEIR SYSTEMS. THE FIRST TWO DEAL STRICTLY WITH ALREADY EXISTING HARDWARE, AND THE THIRD WILL REQUIRE BOTH SOFTWARE AND THE MAKING OF HARDWARE FOR THOSE TECHNICAL PEOPLE OUT THERE. I HOPE THAT YOU FIND THEM INTERESTING AND USEFUL.

YOU MAY ATTEMPT TO INCORPORATE THE FIRST PROGRAM, INTO THE HARDWARE DESIGNED BY RON GRIES ON PAGE 7.

<------------TONES #2------------>

by Matt Petty

I've had a few people that call my BBS ask me how you input from a Hayes Smartmodem or Hayes compatable from XBASIC without getting that nasty little I/O error. Well most people probably know already, but I am going explain how[plus I couldn't think of anything better to write about]. But on top of this, I made an autodial program that runs in XB. I will explain this line for line. First I'm going to tell you how to get around that error. The answer is quite simple:

```
NUM
>100 OPEN
#1:"RS232.LF.EC.TW.BA=xxxx"
!Replace xxx with your baud rate
>110 CR$=CHR$(13):: PRINT
#1:"ATZ";CR$ :: GOSUB 170
>120 PRINT
#1:"ATS11=45S7=20M0V0";CR$ ::
GOSUB 170
>130 ON ERROR 180
>140 PRINT #1:"ATDTxxxxxxx";CR$ ::
GOSUB !Replace x with number to
dial
>150 LINPUT #1:IN$
>160 IF IN$="1" THEN PRINT
"CONNECT!" :: END ELSE 130
>170 FOR DELAY=1 TO 150 :: NEXT
DELAY :: RETURN
>180 ON ERROR 180 :: CALL
ERR(Q,W,E,R):: IF R=150 THEN
RETURN 150 ELSE PRINT "I guess it
has an error!"
```

Those for next delays are just to give the modem enough time to respond to the command.

Lines 130 and 180 solves our problem. If an error occurs it goes to line 180. Line 180 puts the line number of the error into the variable R. Now it checks to see if the error was at the line of the linput, if so, it returns to the linput. Simple.

```
<-------------------------------->
!            AUTO DIAL           !
!               by               !
!           Matt Petty           !
<-------------------------------->
```

This program is fairly simple and was easy to make. It might have a few bugs, so if you find any just call my BBS and tell me.

Explanation:

Lines 1-3: Pre-scan

Line 4:
Clears the screen and changes character and screen color. The on warning next is so it won't have a warning when you press return on a variable input.

Line 5:
If a call init hasn't been preformed, it will go to line 6 and do it, else it will disable the quit key.

Line 7:
If there is an error, goto line 10.

Line 8:
Opens DSK1.DIALFILE. Line 7 made it so that if there is none it gives you the option to set up one.

Line 9:
Inputs all the prestored numbers into the dimension N$.

Line 10:
Gives user option to set up a dail file.

Line 11:
If the user selected to set up a dial file, it gosub's to line 14 which lets him set it up.

Lines 12-13:
Saves dial file to disk at re-runs the program.

Lines 14-20:
Let's the user set up how many numbers he wants to dial, what the numbers are, and how long the modem waits for a corrier.

Lines 21:
Opens the RS232 port and sends the modem register values. Register S11 is the register for how much of a time delay is put between each tone(digit) dialed. I set it at 45, which is fast and accurate. Register S0 is the register that holds the value of how many rings the modem waits for before it answers with a carrier. In this case I set it for 0, so it won't answer. Register S7 holds the value of how many seconds the modem waits for a carrier after dailing a number. You set this.

Line 22:
Clears the screen.

Line 23:
Dials a number and increments the amount of numbers dialed by one.

Lines 24-25:
Displays information on your screen.

Line 26:
Linput's to the RS232c port for a response from the modem.

Line 27:
If the modem connects, it goes to line 31, or if the phone rang it will go to line 30.

Line 28:
Scans the keyboard so if you press 'Q', it will quit and go to the title screen.

Line 29:
Sets a delay, then preforms a NEXT I so if you selected to dial more than one number it will dial the next one. It will continue to preform the NEXT I sequentially until all the numbers are dialed, then it will increment the number of passes, then starts dialing with the first number.

Line 30:
Clears screen, gives alerting sound, then tells the user that the phone rang and gives him the option to keep dialing.

Line 31:
Clears screen, gives alerting sound, tells the user what number number he/she is connected to.

Line 32:
Gives user option to start-up the terminal program installed.

Line 33:
Scans keyboard.

Line 34:
Quits program.

Line 35:
Puts line number that the error was on into variable R. If it equals line number 26 then it returns to it.

Line 36:
This is the 'YES OR NO' input subroutine Lines.

37-38:
Instructions are on those lines.

<-------------------------------->

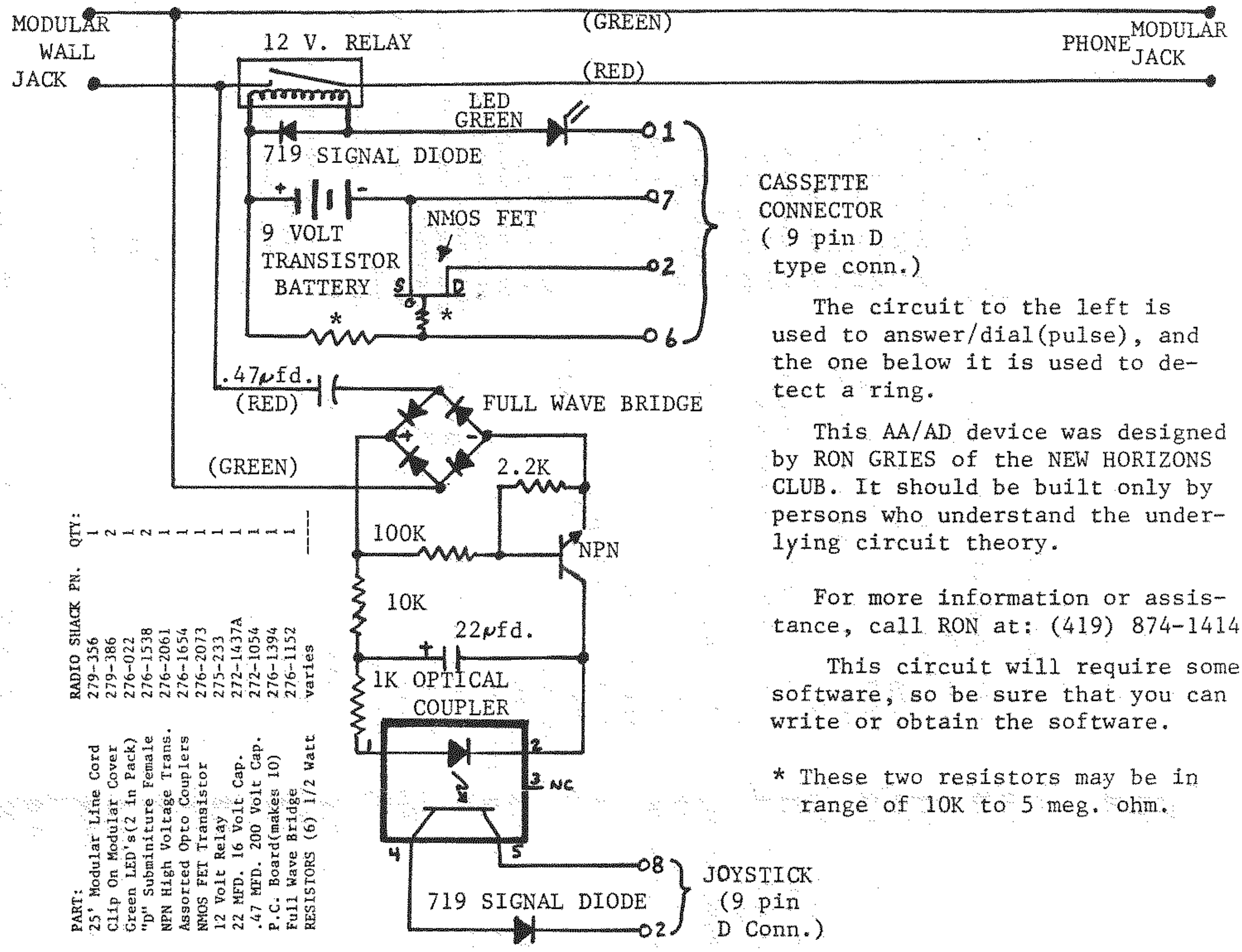
Remember you must have a Hayes or a Hayes compatable modem for this program or it won't work. If you have any suggestions for what I should write about then call me at (714)669-9242.

The above description is for the program on the next page. I've saved it from a newsletter with credit to the author, but I must appologize for having missplaced the name of the Newsletter where this article first appeared. If you use it or find that you can improve on it, you will find Matt Petty's phone number above..... You will note that this page is reduced. You may think that I have failed in my promise to keep the text full sized, well this is full sized as I type it isn't it? No, seriously, the program on the next page is full sized, but it is hard to withdraw in just one month completely from old habits. Please forgive me.

Following Mr. Petty's article, you will find a circuit that was designed by the now very famous RON GRIES of the New Horizons Club, which is used as an auto answer/auto dial device. You will note that he uses the cassette port for the answer/dial section of the circuit, and the joystick input, for the ring detector.

```
1 GOTO 4
2 CALL ERR :: CALL SOUND :: CALL CLEAR :: CALL INIT :: CALL KEY :: CALL COLOR ::
 CALL SCREEN :: CALL LOAD :: Q,W,A,B,I,R,RR,C,PASS,CO=0 :: DIM N$[9]:: CA$,CO$="
"
3 !@P-
4 CALL CLEAR :: CALL SCREEN[6]:: FOR I=0 TO 14 :: CALL COLOR[I,16,2]:: NEXT I ::
 ON WARNING NEXT
5 ON ERROR 6 :: CALL LOAD[-31806,16]:: GOTO 7
6 CALL INIT :: CALL LOAD[-31806,16]
7 ON ERROR 10
8 OPEN #1:"DSK1.DIALFILE",RELATIVE,INTERNAL,FIXED 18 :: INPUT #1,REC 0:A,B
9 FOR I=1 TO A :: INPUT #1,REC I:N$[I]:: NEXT I :: CLOSE #1 :: ON ERROR 35 :: GO
TO 14
10 R=10 :: CA$="Set up dial file[Y/N]:" :: GOSUB 36 :: IF CA$="N" THEN RUN 1
11 RR=1 :: GOSUB 14
12 DISPLAY AT[24,1]:"Saving..." :: OPEN #2:"DSK1.DIALFILE",RELATIVE,INTERNAL,OUT
PUT,FIXED 18 :: PRINT #2,REC 0:A,B
13 RR=0 :: FOR I=1 TO A :: PRINT #2,REC I:N$[I]:: NEXT I :: CLOSE #2 :: RUN
14 CALL CLEAR :: DISPLAY AT[5,1]:"0 to set up dial file":"How many numbers[1-9]:
"&STR$[A]
15 ACCEPT AT[6,23]SIZE[-1]:A :: IF A=0 THEN 10
16 FOR I=1 TO A
17 DISPLAY AT[I+7,1]:"Number "&STR$[I]&":"&N$[I]:: ACCEPT AT[I+7,10]VALIDATE["-0
987654321"]SIZE[-20]:N$[I]:: IF N$[I]="" THEN 20
18 NEXT I
19 DISPLAY AT[22,1]:"Time delay:"&STR$[B]:: ACCEPT AT[22,12]VALIDATE["0987654321
"]SIZE[-2]:B :: IF B<4 THEN 19
20 R=24 :: CA$="Are these correct[Y/N]:" :: GOSUB 36 :: IF CA$="N" THEN 14 ELSE
IF RR=1 THEN RETURN
21 OPEN #1:"RS232.LF.CR" :: PRINT #1:"ATZ";CHR$[13]:: FOR I=1 TO 200 :: NEXT I :
: CA$="ATS11=45S0=0S7="&STR$[B]&"V0M0" :: PRINT #1:CA$;CHR$[13]:: FOR I=1 TO 200
 :: NEXT I
22 CALL CLEAR
23 FOR I=1 TO A :: CA$="ATDT"&N$[I]:: PRINT #1:CA$;CHR$[13]:: C=C+1
24 DISPLAY AT[6,1]:"Dialing:"&N$[I]:: DISPLAY AT[8,1]:"Calls made:"&STR$[C]:: DI
SPLAY AT[10,1]:"Passes made:"&STR$[PASS]
25 DISPLAY AT[12,1]:"Times connected:"&STR$[CO]:: IF CO>0 THEN DISPLAY AT[14,1]:
"You have been connected on:":CO$
26 LINPUT #1:CA$
27 IF CA$="1" THEN 31 ELSE IF CA$="2" THEN 30
28 CALL KEY[0,Q,W]:: IF W THEN IF CHR$[Q]="Q" THEN 34
29 FOR Q=1 TO 200 :: NEXT Q :: NEXT I :: PASS=PASS+1 :: GOTO 23
30 CALL CLEAR :: CALL SOUND[2000,110,0]:: DISPLAY AT[10,8]:"TELEPHONE!!!" :: R=1
5 :: CA$="Continue dialing?" :: GOSUB 36 :: IF CA$="Y" THEN 22 ELSE 30
31 CALL CLEAR :: CALL SOUND[2000,110,0]:: DISPLAY AT[2,1]:"Connected to >"&N$[I]
32 DISPLAY AT[10,1]:"PRESS 'P' for term":"Any other key to Return" :: CO$=CO$&N$
[I]:: CO=CO+1
33 CALL KEY[0,Q,W]:: IF W=0 THEN 33 ELSE IF CHR$[Q]="P" THEN 37 ELSE 29
34 CALL LOAD[-31961,51]:: END
35 ON ERROR 35 :: CALL ERR[Q,W,RR,R]:: IF R=26 THEN RETURN 26 ELSE STOP
36 CA$=CA$&"Y" :: DISPLAY AT[R,1]:CA$ :: ACCEPT AT[R,LEN[CA$]]VALIDATE["YN"]SIZE
[-1]:CA$ :: IF CA$="" THEN 21 ELSE RETURN
37 CALL LOAD["DSKx.yyyyy"]
   Replace x with drive #
   Replace y with terminal program name
38 CALL LINK["xxxxx"]
   Replace x with terminal program like[if there is one]
```

AUTO ANSWER/AUTO DIAL INTERFACE



PART:	RADIO SHACK PN.	QTY:
25' Modular Line Cord	279-356	1
Clip On Modular Cover	279-386	2
Green LED's(2 in Pack)	276-022	1
"D" Subminiture Female	276-1538	2
NPN High Voltage Trans.	276-2061	1
Assorted Opto Couplers	276-1654	1
NMOS FET Transistor	276-2073	1
12 Volt Relay	275-233	1
22 MFD. 16 Volt Cap.	272-1437A	1
.47 MFD. 200 Volt Cap.	272-1054	1
P.C. Board(makes 10)	276-1394	1
Full Wave Bridge	276-1152	1
RESISTORS (6) 1/2 Watt	varies	-----

The circuit to the left is used to answer/dial(pulse), and the one below it is used to detect a ring.

This AA/AD device was designed by RON GRIES of the NEW HORIZONS CLUB. It should be built only by persons who understand the underlying circuit theory.

For more information or assistance, call RON at: (419) 874-1414

This circuit will require some software, so be sure that you can write or obtain the software.

* These two resistors may be in range of 10K to 5 meg. ohm.

POWER SUPPLIES AND KEYBOARDS

Today is MAY 8, 1986, and at this very minute you can get both TI KEYBOARDS and TI SWITCHING POWER SUPPLIES from :

ELECTRONICS PARTS OUTLET
2275 So. FEDERAL HWY.
Del Ray Beach, FL
(305) 265-1206 ,7 33444

I just got off the phone, and it sounds like they have a lot of them in stock. The price is right, if you tell them you are answering their advertisement in the COMPUTER SHOPPER, MAY issue, Page 47, lower right hand corner. The price that is advertised there is : KEYBOARDS-------- $2.95
POWER SUPPLIES-- $1.25
.... even better than most of us got them from RADIO SHACK! I hope that this helps someone find these very useful items. Also, if they are out you may try their other two stores, PHOENIX, AZ (602) 375-0181 ,2 or the store in SCOTTSDALE, AZ , (602) 941-9377, 9357, 9328, 9329.

WAITING TO BUY A DSDD DISK DRIVE ??? BETTER HURRY!!!

With the YEN increasing against the dollar, we have noticed an increase in both diskette drives, and components(IC Chips). If you try to order a TEAC 55B, for instance, you will more than likely pay $10. to $20. dollars more than the price advertised in whatever media you see the ad in. So if you are serious, you better dig a little deeper into your pocket now, for there is not any indication of a down turn in prices. GOOD LUCK! John F. Willforth WPUG.

Convert BASIC to Extended BASIC

John Behnke, of the Chicago TI Users Group, has a program that is a real time-saver for those who want to convert BASIC programs into Extended BASIC programs. Called VDP Utility II, the program was published in the Chicagoans newsletter, Chicago TImes. Since the VDPU II must be merged with the BASIC program you want to convert, it is recommended that you not change the line-numbering. Save the program in MERGE format. After loading a BASIC program in Extended BASIC, merge the VDPUTIL2 program into it and save the two programs as one. Then run it.

The program, requires Extended BASIC, a disk system and memory expansion.

```
32700 !"VDP UTILITY II"
32701 !BY JOHN BEHNKE
32702 !
32703 !CHICAGO, ILL.
32704 !WILL CONVER ANY BASIC
32705 !PROGRAM TO X-BASIC
32706 !DIRECTIONS: LOAD BASI
C
32707 !PROGRAM INTO X-BASIC.
32708 !THEN INPUT:
32709 !"MERGE DSK1.VDPUTIL2"
32710 !WHEN FINISHED, RE-SAV
E
32711 !BASIC PROGRAM. THE RE
SULTING
32712 !PROGRAM WILL RUN IN
32713 !X-BASIC
32714 SUB VDPUTIL2
32715 CALL CLEAR :: CALL INI
T :: CALL LOAD(8196,63,232)
32716 CALL LOAD(16360,80,79,
75,69,82,32,38,12,80,79,75,6
9,86,32,37,164,80,69,69,75,8
6,32,37,36)
32717 CALL LOAD(9491,100)
32718 CALL LOAD(9508,2,224,3
7,20,3,0,0,0,2,0,0,100,200,0
,37,18,4,192,2,1,0,1,4,3,2,3
2,12,4,32)
32719 CALL LOAD(9536,32,24,1
8,184,192,32,131,74,2,1,37,0
,208,160,131,18,9,130,2,34,2
55,255,4,32,32,44)
32720 CALL LOAD(9562,4,197,2
09,34,36,255,9,132,19,21,4,1
95,60,224,37,18,200,5,131,76
,200,5,131,78,200,5)
32721 CALL LOAD(9588,131,80,
2,5,64,0,161,68,2,131,0,1,17
,6,2,5,65,0,161,67,6,196,200
,4,131,76)
32722 CALL LOAD(9614,200,5,1
31,74,4,192,192,66,5,129,4,3
7,254)
32723 CALL LOAD(9636,2,224,3
7,20,3,0,0,0,4,192,2,1,0,1,2
00,1,37,18,4,32,32,12,4,32,3
2,24,18,184)
32724 CALL LOAD(9664,200,32,
131,74,37,0,184,32,131,18,37
,19,2,3,0,2)
32725 CALL LOAD(9680,4,192,1
92,67,4,32,32,12,4,32,32,24,
18,184,216,224,131,75,37,0,5
,131,136,3)
32726 CALL LOAD(9704,37,18,2
2,242,192,32,37,0,2,1,37,2,1
92,131,2,34,255,254,4,32,32,
36)
32727 CALL LOAD(9726,4,192,2
16,0,131,124,2,224,131,224,4
,96,0,112)
32728 CALL LOAD(9740,3,0,0,0
,4,192,2,1,0,1,4,32,32,12,20
0,32,131,74,37,18,2,1,0,2,4
32,32,12,4,32)
32729 CALL LOAD(9770,32,24,1
8,184,192,32,131,74,208,32,3
7,19,4,32,32,48,4,91)
32730 CALL LOAD(8194,39,04)
32731 SUBEND
32732 SUB CHAR(A,A$):: L=LEN
(A$)
32733 A$=A$&RPT$("0",16-L)
32734 FOR I=1 TO 16 STEP 2
32735 A1$=SEG$(A$,I,1)
32736 A2$=SEG$(A$,I+1,1)
32737 IF A1$<":" THEN A1=VAL
(A1$)*16 ELSE A1=(ASC(A1$)-5
5)*16
32738 IF A2$<":" THEN A1=A1+
VAL(A2$)ELSE A1=A1+ASC(A2$)-
55
32740 CALL LINK("POKEV",767+
8*A+(I+1)/2,A1)
32741 NEXT I
32742 SUBEND
32743 SUB COLOR(A,B,C)
32744 CALL LINK("POKEV",2063
+A,(B-1)*16+C-1)
32745 SUBEND
```

THIS PROGRAM IS A REPRINT FROM "ENTER", BY TRAVIS WORKS OF RINGOLD, GA. IT IS WRITTEN IN TI BASIC.

PRESS A NUMBER KEY AND THEY CHANGE STEPS. HOLD ONE KEY DOWN, FOUR KEEP RIGHT ON DANCING.

BREAKDANCING

```
10 RANDOMIZE
20 GOSUB 250
30 PRINT "BREAKDANCING!!!"
40 PRINT "HUMAN OR COMPUTER
CONTROL???"
50 INPUT CON$
55 CALL CLEAR
60 IF CON$="HUMAN" THEN 120
70 BD=INT(RND*5)+153
80 CALL KEY(0,W,E)
90 IF E=1 THEN 120
100 GOSUB 180
110 GOTO 70
120 CALL KEY(0,BD,N)
130 IF N=0 THEN 120
140 IF BD=32 THEN70
150 BD=BD+102
160 GOSUB 180
170 GOTO 120
180 CALL VCHAR(12,10,BD-(INT
(RND*2))+1)
190 CALL VCHAR(12,12,BD-(INT
(RND*2))+1)
200 CALL VCHAR(12,16,BD)
210 CALL VCHAR(12,20,BD+(INT
(RND*2))+1)
220 CALL VCHAR(12,22,BD+(INT
(RND*2))+1)
230 RETURN
240 GOTO 120
250 REM CHARACTERS
260 CALL CHAR(151,"000000000
4B84438")
270 CALL CHAR(152,"00008244З
8383854")
280 CALL CHAR(153,"0010FE383
84482")
290 CALL CHAR(154,"8090FC3A3
9484808")
300 CALL CHAR(155,"000000000
0847936")
310 CALL CHAR(156,"00107CBA7
C281808")
320 CALL CHAR(157,"142424783
8")
330 CALL CHAR(158,"41493E1C1
C2214")
340 CALL CHAR(159,"40281E1D1
41414")
350 RETURN
```

Bubble sort in Forth

By HOWARD A. ARNOLD

Here is a Forth procedure for doing a bubble sort on a disk resident file. You may incorporate it in your file management or data base programs or use it independently to do an alphabetic sort on a series of entries in order to test your Forth wings. Though written in TI-Forth, it should be easily convertible to other dialects.

In the interest of keeping the procedure on a single Forth screen, I have omitted on-screen comments. These comments and explanations are offered here in case the code isn't entirely self-explanatory.

```
SCR #88
  0 ( BUBBLE SORT)
  1 1 VARIABLE FLAG   16 VARIABLE MAXRECS  0 VARIABLE #RECORD
  2 64 CONSTANT R-LENGTH   30 CONSTANT W-LENGTH  78 CONSTANT FILES
  3 1024 R-LENGTH / CONSTANT REC/BLK        0 VARIABLE TEMP 80 ALLOT
  4 : 2DUP OVER OVER ;  : NOT 0= ; : MOVE 2 / MOVE ;
  5 : -TEXT 2DUP + SWAP DO DROP 2+ DUP 2- @ I @ - DUP IF DUP ABS
  6   / LEAVE THEN 2 +LOOP SWAP DROP ;
  7 : RECORD
  8         #RECORD @ REC/BLK /MOD FILES + BLOCK SWAP R-LENGTH * + ;
  9 : SWITCH RECORD DUP -1 #RECORD +! RECORD DUP TEMP R-LENGTH MOVE
 10   R-LENGTH MOVE TEMP SWAP R-LENGTH MOVE 1 FLAG ! UPDATE ;
 11 : SORT MAXRECS @ 1- 0 DO I #RECORD ! RECORD W-LENGTH 1 #RECORD
 12   +! RECORD -TEXT 0 > IF SWITCH ENDIF #RECORD @ . LOOP ;
 13 : DOSORT BEGIN FLAG @ IF 0 FLAG ! SORT ENDIF FLAG @ NOT UNTIL 1
 14   FLAG ! ;
 15 : WRITE #RECORD ! RECORD R-LENGTH EXPECT UPDATE ;
```

Line 1: Variable declaration

FLAG is used to indicate whether a postition switch has been made of any records within a given pass through the file.

MAXRECS stores the total number of records to be sorted. It is initialized here to a value if constant, or may be changed by your application program to retain the number of records to be sorted if that number is changed in the parent program.

#RECORD is the number (between 0 and MAXRECS) of the record currently being accessed.

Lines 2-3: Constant declarations

R-LENGTH is the maximum length of the strings being sorted.

W-LENGTH is the length of the string to be examined for alphabetic sorting.

FILES is the number of the Forth screen on which the first record is stored in the file to be sorted.

REC/BLK is the number of records which will fit on a given screen. This is calculated based on the R-LENGTH declared.

TEMP is a pad set aside for temporarily storing a string whose position is to be switched during the sort. It must be allotted a length at least as great as R-LENGTH.

Lines 4-6: Compatibility definitions

These definitions, mostly taken from Leo Brodie's "Starting Forth," provide compatibility between the FIG-Forth standards and the TI-Forth language. Note that the definition for MOVE, since it uses the previously defined word MOVE within its own definition, should be loaded only once. In other words, FORGET FLAG each time you reload this screen to avoid unexpected results.

Line 7: RECORD definition

This word provides the address of the buffer storing the string currently being accessed.

Line 9: SWITCH definition

This word switches the position of two strings in adjacent locations in the file. The string from the lower location is stored in TEMP, the next higher string stored in its place and the TEMP string is written to the higher of the two locations. The buffer is then marked with UPDATE so that it can be subsequently FLUSHED to the disk file.

Line 11: SORT definition

This word examines two adjacent strings of length W-LENGTH using the word -TEXT defined on line 6. If -TEXT returns a 1, indicating that the strings are out of alphabetic order, the word SWITCH defined above is called, inverting the two positions.

Line 13: DOSORT definition

This is the final USER WORD which initiates the sort. It keeps track of the status of FLAG and continues calling SORT until a complete examination of the record is accomplished without the necessity for an alphabetical swap.

Line 15: WRITE definition

In order to build a file to test the program, the word WRITE is provided. Use WRITE like this:

n WRITE (ENTER) xxxxxxxxxx (ENTER)

where n is the record of the string to be entered, and xxxxxxxxxx is the string itself.

Be sure to change variable MAXRECS to agree with the number of records to be sorted with the command n MAXRECS ! (after loading the screen); n in this case being the total number of records to be sorted.

You'll find this to be a surprisingly fast bubble sort, I suppose because of the inherent speed of Forth.

Enjoy!

Arnold is a retired engineering manager of AT&T Technologies. He now does freelance writing and consulting both in computer programming and manufacturing technology. He can be reached at 210 Beech Valley Rd Lewisville, NC 27023.
(919) 945-5348

TK-WRITER FIX
By Jim McGarvie from San Diego's TIBBS: via DELEWARE VALLEY U.G.

To avoid data loss if you should accidently use the Show Directory (SD) command while using TK-WRITER try the following modification:

Use a sector editor utility to change byte >14 (decimal 20) of the third sector of EDITA1 from >53 to >20. Then the "SD" beeps instead of destructing.

Another suggestion from Woodrow Wilson, was to call the TK-WRITER via:

```
10 RUN "DSK1.TK-WRITER"
20 END
```
then add line 265 in TK-WRITER to call
```
265 RUN "DSK1.CAT"
```
to bring up an appropriate catalog. (You have to supply the catalog program

THIS PAGE IS INTENDED FOR THE BEGINNER IN THE ART OF ASSEMBLY PROGRAMMING. THE PROGRAMS ARE NOT LONG, AND SHOULD NOT LEND THEMSELVES TO A LOT OF ERRORS, BUT THE FACT THAT YOU WILL HAVE TO ENTER AND COMPILE THEM, WILL BE GOOD TRAINING. IF YOU HAVE A PROBLEM, I'M SURE THAT A CALL TO GENE KELLY OR CLYDE COLLEDGE WILL CLEAR IT UP QUICKLY. GOOD LUCK!

No. NJ 99'er Users Group Newsletter

```
::::::::::::::::::::::::::::::::::::::::::   ****************************
::                                      ::   *     TRUE LOWER CASE      *
:: TI SOURCE's (tm)  Assembly Corner    ::   * from hcm vol.4.4 pg.79   *
::                                      ::   * modified by tony gomes   *
::::::::::::::::::::::::::::::::::::::::::   ****************************

*
AL    DATA >0000,>0070,>0838,>4874    Lower case charactor set.
      DATA >0040,>4078,>4444,>4478
      DATA >0000,>0038,>4440,>4438
      DATA >0004,>043C,>4444,>443C
      DATA >0000,>0038,>447C,>403C
      DATA >0018,>2420,>7020,>2020
      DATA >0000,>0438,>4438,>047C
      DATA >0040,>4078,>4444,>4444
      DATA >0010,>0030,>1010,>1038
      DATA >0008,>0018,>0808,>4830
      DATA >0040,>4048,>5070,>4844
      DATA >0030,>1010,>1010,>1038
      DATA >0000,>0078,>5454,>5454
      DATA >0000,>0058,>2424,>2424
      DATA >0000,>0038,>4444,>4438
      DATA >0000,>0078,>4478,>4040
      DATA >0000,>0038,>443C,>0406
      DATA >0000,>0058,>6440,>4040
      DATA >0000,>003C,>4038,>0478
      DATA >0010,>3810,>1010,>1408
      DATA >0000,>0048,>4848,>4824
      DATA >0000,>0044,>4428,>2810
      DATA >0000,>0044,>5454,>5424
      DATA >0000,>0044,>2810,>2844
      DATA >0000,>0044,>2418,>1060
      DATA >0000,>007C,>0810,>207C
LOW   MOV 11,10      save link to basic.
      LI 0,>0608     address of ascii 97 in vdp ram.
      LI 1,AL        address of charactor codes.
      LI 2,208       208 (26*8) bytes to write
      BLWP >2024  multiple byte write (in x-basic.>6028 in e/a)
      B          return to x-basic (or basic in e/a,mini-mem)

      AORG  >83C4    address of user defined interrupt routine. the data lo
      DATA  LOW      statement is the address of the program. Once loaded tl
*                    program is automatically implemented. There is no need
*                    to call link. Call init will not stop this routine. you
*                    must reset the computer.
*
*
*This version is for X-Basic.It gives you true lower case in imediate moc
*as well as in programs. Since it is an interrupt routine, it shares the
*cpu with the program. It will slow down program execution a bit but, not
*too much. To use it with E/A or Mini-Mem basic, change line 42 to:
*  BLWP >6028. To make it a non-interrupt routine, one that can only be
*linked to a program(call link("low"),delete lines 45 and 46. Also add DE
*LOW at the beginning.
*Assemble using OBJECT FILE NAME:DSKx.LOWINT
*                 OPTIONS: R

::::::::::: ::::::::::::::::::::::::
```

MORE ON THE LOAD INTERRUPT SWITCH

by G A R Y B R O W N (from the J*U*G*S Newsletter)

Editor: If you havent already installed a LOAD INTERRUPT switch, all that you need to do is install a momentary contact switch between the 13th pin and any of the ground pins (21,23,25, or 27) of the edge connector of your TI-99 console CPU board, or the corresponding pins in a device that plugs into this connector on the right side of the console.(speech synthesizer, fire hose connector, stand alone memory, RS232, or disk unit). Neither I or the author of this article can assume any responsibility for any damage that you may do to your equiptment.

Assemble this source code:

```
AORG >FFFC
DATA >A000,B08E END
```

Load SBUG
Load the interrupt program you have assembled above.
Press Function =(QUIT)
Do not turn console or PEB off.
Insert a cartridge and start it running.
Press the interrupt switch and you will have the SBUG title screen.

Follow the instructions and go into address >6000 thru >8000 to find the program in these addresses. Then you can disassemble these addresses.

Thank You GARY, from the WPUG!

```
***************************
*                         *
*  ASSEMBLY LANGUAGE      *
*  ROUTINE TO CHANGE      *
*  THE TEXT AND SCREEN    *
*  COLORS IN X-BASIC      *
*  FOR PROGRAMMING        *
*                         *
*  BY LARRY BENTLEY       *
*  9/29/84                *
*                         *
***************************
       DEF  COLOR
*
* SET EQUATES FOR XBASIC
VMBW   EQU  >2024
VWTR   EQU  >2030
C      DATA >F4F4,>F4F4,>F4F4,>F4F4  *
SET 8 BYTES ASIDE TO LOAD
COLOR  LI   R0,>07F4              *
CHAR COLORS. ALSO LOAD VDP
VWTR                      *
R   TH SCREEN BKGRD COLOR
       LI   R0,2048                     *
START LOADING CHARSET COL-
       LI   R1,C                        *
ORS
       LI   R2,8
VMBW
       LI   R0,2056
VMBW
       LI   R0,2064
VMBW
       LI   R0,2072
VMBW
       B    *R11                        *
RETURN TO MAIN ROUTINE
       AORG >83C4                       *
DURING LOADING, MOVE TO >83C4
       DATA COLOR                       *
PUT THE ADDRESS REPRESENTED BY
       END                              *
COLOR INTO >83C4

*-----------------------  -----------
----------*
```

GETTING THE MOST FROM YOUR CASSETTE SYSTEM

BY MICKEY SCHMITT

NUMBER 1

GETTING STARTED

BEFORE YOU TRY TO DO ANYTHING WITH A CASSETTE SYSTEM YOU NEED TO START WITH THE RIGHT EQUIPMENT. THERE ARE MANY DIFFERENT MODELS OF STANDARD CASSETTE RECORDERS AVAILABLE WHICH WILL WORK WITH YOUR TI COMPUTER. (BESIDES THE OFFICIAL TI PROGRAM RECORDER). HOWEVER, FOR BEST OPERATION AND ALOT LESS MENTAL AGGRAVATION, YOU SHOULD USE A CASSETTE RECORDER WITH THE FOLLOWING FEATURES:

1. VOLUME CONTROL: FOR BEST RESULTS THIS SHOULD BE SET BETWEEN MID-RANGE AND MAXIMUM SETTINGS.
2. TONE CONTROL: FOR BEST RESULTS THIS SHOULD ALSO BE SET BETWEEN MID-RANGE AND MAXIMUM SETTINGS.
3. MICROPHONE JACK: THIS JACK IS NEEDED IN ORDER TO RECEIVE INFORMATION FROM YOUR COMPUTER.
4. EARPHONE OR EXTERNAL SPEAKER JACK: THIS JACK IS NEEDED IN ORDER TO SEND INFORMATION TO YOUR COMPUTER.
5. REMOTE JACK: THIS JACK MAKES IT POSSIBLE FOR YOUR COMPUTER TO CONTROL YOUR CASSETTE RECORDER'S DRIVE MOTOR - THUS YOUR TAPE RECORDER WILL RUN BY PRESSING THE 'ENTER' KEY ON YOUR COMPUTER CONSOLE.
6. DIGITAL TAPE COUNTER: THIS IS A VERY IMPORTANT FEATURE AS IT WILL SAVE YOU ALOT OF UNNECESSARY AGGRAVATION. THIS FEATURE ENABLES YOU TO EASILY LOCATE THE CORRECT TAPE POSITION OF YOUR PROGRAM OR DATA FILE. THIS IS ESPECIALLY USEFUL WHEN YOU WANT TO STORE MORE THAN ONE PROGRAM ON THE SAME SIDE OF THE CASSETTE TAPE.

NEXT, YOU WILL NEED TO HAVE THE TI CASSETTE INTERFACE CABLE WHICH IS USED TO CONNECT YOUR RECORDER TO YOUR COMPUTER. ALTHOUGH, THIS CABLE COMES WITH THE OFFICIAL TI PROGRAM RECORDER, IT MUST BE PURCHASED SEPARATELY IF YOU ARE USING ANOTHER TYPE OF CASSETTE RECORDER. IF YOU ARE HAVING TROUBLE FINDING THIS CABLE I WOULD SUGGEST TRYING THE COMPUTER BUG (412-882-3374) 5075 CLAIRTON BLVD. PITTSBURGH, PA 15236. THE FOLLOWING INSTRUCTIONS WILL GUIDE YOU THROUGH THE PROCESS OF CONNECTING YOUR CASSETTE RECORDER TO YOUR COMPUTER USING THE TI INTERFACE CABLE:

1. LOCATE THE NINE-PIN PLUG AT ONE END OF THE CASSETTE RECORDER INTERFACE CABLE. INSERT THIS PLUG FIRMLY INTO THE JACK ON THE RIGHT REAR OF THE COMPUTER.
2. LOCATE THE SET OF THREE PLUGS AT THE OTHER END OF THE CABLE. THE WIRES THAT LEAD TO THESE PLUGS ARE COLOR-CODED: RED-WHITE-BLACK.
3. LOCATE THE JACKS LABELED MIC, EAR (OR EXTERNAL SPEAKER) AND REM ON YOUR CASSETTE RECORDER.
4. INSERT THE PLUG WITH THE RED WIRE INTO THE RECORDER'S MICROPHONE JACK (LABELED MIC).
5. INSERT THE PLUG WITH THE WHITE WIRE INTO THE RECORDER'S EARPHONE (OR EXTERNAL SPEAKER) JACK (LABELED EAR).
6. INSERT THE PLUG WITH THE BLACK WIRE INTO THE RECORDER'S REMOTE JACK (LABELED REM).

THAT'S ALL THERE IS TO IT! YOUR CASSETTE SYSTEM IS NOW READY TO GO. NEXT MONTH'S TOPIC WILL BE LOADING AND SAVING PROGRAMS.

IF YOU NEED ANY HELP IN GETTING YOUR CASSETTE SYSTEM STARTED - JUST GIVE ME A CALL (412-335-0163) AND I'LL TRY TO HELP.

MICKEY SCHMITT

This banner program comes to us from the Atlanta 99/4A Computer Users Group (A9CUG) Call Newsletter. There were no credits given as to who is the author of the program. It will print out a banner "sideways" on your printer in varying letter heights. The DATA statements at the end set up the block style for the letters. If you are not satisfied with the output you can modify the letters to suit your taste.

```
100 DIM A(471),BIN(6)
110 CALL CLEAR
120 PRINT "       GETTING RE
ADY                PLEASE WA
IT"
130 FOR I=0 TO 464 STEP 8
140 FOR J=0 TO 7
150 IF (J<>0)*(J<>7)THEN 180
160 A(I+J)=0
170 GOTO 190
180 READ A(I+J)
190 NEXT J
200 NEXT I
210 A(80)=8
220 A(87)=8
230 A(367)=63
240 A(447)=63
250 FOR I=0 TO 6
260 BIN(I)=2^I
270 NEXT I
280 CALL CLEAR
290 PRINT TAB(11);"*BANNER*"
300 PRINT
310 PRINT "HOW MANY LETTERS
CAN YOUR"
320 INPUT "PRINTER PRINT ON
A LINE?":CL
330 IF (CL>=7)*(CL<=136)THEN
 380
340 PRINT
350 PRINT "I DON'T THINK THA
T'S RIGHT"
360 PRINT "PLEASE CHECK YOUR
 MANUAL"
370 GOTO 310
380 CALL CLEAR
390 PRINT "HOW TALL DO YOU W
ANT THE"
400 PRINT "LETTERS IN YOUR B
ANNER?"
410 PRINT
420 PRINT "(1=SHORTEST; ";ST
R$(INT(CL/7));"=TALLEST)";
430 INPUT CC
440 IF (CC>=1)*(CC<=INT(CL/7
))THEN 470
450 PRINT
460 GOTO 420
470 CALL CLEAR
480 PRINT "WHAT CHARACTER SH
ALL I USE"
490 PRINT "TO COMPOSE THE LE
TTERS OF"
500 PRINT "YOUR BANNER"
510 PRINT
520 INPUT "(E.G., *, $, #)?"
:CH$
530 IF CH$="" THEN 470
540 CH$=SEG$(CH$,1,1)
550 CALL CLEAR
560 PRINT "PLEASE ENTER THE
MESSAGE YOU";
570 PRINT "WANT PRINTED ON Y
OUR BANNER";
580 PRINT "DO NOT USE LOWERC
ASE"
590 PRINT "LETTERS OR COMMAS
"
600 PRINT
610 INPUT MESSAGE$
620 PRINT
630 PRINT "WHEN YOUR PRINTER
 IS READY,"
640 PRINT "PLEASE PRESS ANY
KEY"
650 CALL KEY(3,K,S)
660 IF S=0 THEN 650
670 REM --USE 680 OPEN #1:"R
S232" FOR SERIAL PRINTER--
680 OPEN #1:"PIO"
690 FOR I=1 TO LEN(MESSAGE$)
700 PNTR=(ASC(SEG$(MESSAGE$,
I,1))-32)*8
710 IF (PNTR>=0)*(PNTR<=464)
THEN 730
720 PNTR=0
730 FOR J=PNTR TO PNTR+7
740 LN$=""
750 V=A(J)
760 FOR K=6 TO 0 STEP -1
770 IF V<BIN(K)THEN 810
780 V=V-BIN(K)
790 C$=CH$
800 GOTO 820
810 C$=" "
820 FOR L=1 TO CC
830 LN$=C$&LN$
840 NEXT L
850 NEXT K
860 FOR K=1 TO (CC+1)/2
870 PRINT #1:LN$
880 NEXT K
890 NEXT J
900 NEXT I
910 CLOSE #1
920 CALL CLEAR
930 PRINT "YOUR BANNER IS FI
NISHED"
940 PRINT
950 PRINT "PRESS <P> TO PRIN
T ANOTHER"
960 PRINT "BANNER OR <E> TO
END"
970 CALL KEY(3,K,S)
980 IF K=ASC("P")THEN 380
990 IF K<>ASC("E")THEN 970
1000 END
1010 DATA 0,0,0,0,0,0,0,0,61
,61,0,0
1020 DATA 0,56,0,0,56,0,18,6
3,18,18,63,18
1030 DATA 18,58,107,107,46,3
6,51,54,12,24,51,35
1040 DATA 6,47,121,93,118,39
,0,0,52,56,0,0
1050 DATA 0,0,30,63,51,33,33
,51,63,30,0,0
1060 DATA 42,62,28,28,62,42,
8,8,62,62,8,8
1070 DATA 0,0,13,14,0,0,8,8,
8,8,8,8
1080 DATA 0,0,3,3,0,0,3,6,12
,24,48,32
1090 DATA 30,63,37,41,63,30,
1,17,63,63,1,1
1100 DATA 17,51,39,45,57,17,
34,35,41,61,55,34
1110 DATA 6,14,26,63,63,2,58
,59,41,41,47,38
1120 DATA 30,63,41,41,47,6,3
2,35,39,44,56,48
1130 DATA 22,63,41,41,63,22,
16,57,41,43,62,28
1140 DATA 0,0,54,54,0,0,0,0,
109,110,0,0
1150 DATA 0,8,28,54,99,65,18
,18,18,18,18,18
1160 DATA 65,99,54,28,9,0,16
,48,37,45,56,16
1170 DATA 30,63,33,45,61,29,
15,31,50,50,31,15
1180 DATA 63,63,41,41,63,22,
30,63,33,33,51,18
1190 DATA 63,63,33,51,30,12,
63,63,41,41,41,33
1200 DATA 63,63,40,40,40,32,
30,63,33,37,39,39
1210 DATA 63,63,8,8,63,63,33
,33,63,63,33,33
1220 DATA 2,3,1,1,63,62,63,6
3,12,30,51,33
1230 DATA 63,63,1,1,1,1,63,6
3,24,12,24,63
1240 DATA 63,63,28,14,63,63,
30,63,33,33,63,30
1250 DATA 63,63,36,36,60,24,
30,63,33,34,63,29
1260 DATA 63,63,36,38,63,25,
16,57,41,41,47,6
1270 DATA 32,32,63,63,32,32,
63,63,1,1,63,63
1280 DATA 60,62,3,3,62,60,63
,63,6,12,6,63
1290 DATA 51,63,12,12,63,51,
48,56,15,15,56,48
1300 DATA 35,39,45,57,49,33
```

++++++++++++++++++++++++++++

ARE THERE ANY "HAMS" OUT THERE?...................

If you are a ham, then you really should take a look at MARTY KROLL'S "MORSETUTOR" program. See back of this newsletter. With it, you can practice code up to 90 words a minute. It will check your speed, and automatically adjust to a slightly lower speed if you are not up to it.

The program is also capable of outputting to the key input of your radio, by using the cassette motor output on the cassette port.

The program will run from cassette and requires only 32K additionally (a good prospect for the 32K internal expansion)

The program is written in xbasic, with assembly routines for speed. You can get it from MARTY at the address shown on the back of this newsletter.

John F. Willforth

CASSETTE COVER MAKER

Curtis Alan Provance
New Hampshire 99er's User Group

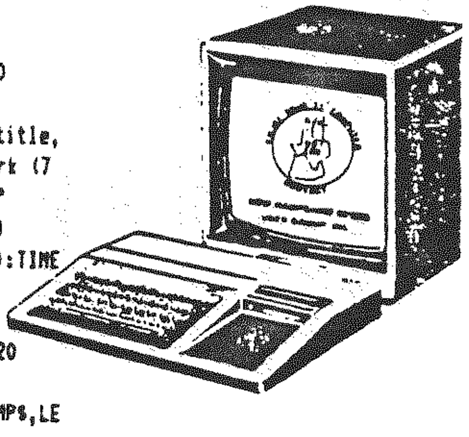
Your recent surveys asked for more BASIC and XBASIC material, so here it is! The following program makes a cover for your cassettes. It can print 14 items on each of two sides. You may also enter tape counter data and time information. I tried to write this in such a way that you can modify it to suit your needs. If you want to enter lower case text, insert the following line:

```
545 CALL KEY(5,K,S)
```

Have fun and experiment! There are many areas that can be improved; I was mainly interested in presenting an easily understood routine. Don't be afraid of using long variable names. If you run out of room, you can go back and shorten them. However, if you go back to a program months after you wrote it, the chances of remembering what's going on are greater if you used good variable names.

```
100 DIM TOTAL(1),NAME$(1,14)
,COUNT$(1,14),TIME$(1,14)
110 DEF CENTER$(X$)=":"&SEG$
(BLANK$,1,32-LEN(X$)/2)&X$&S
EG$(BLANK$,33+LEN(X$)/2,33)&
":"
120 EDGE$=":----------------
----------------------------
--------------------:"
130 BLANK$="


                   "
140 BORDER$=":"&BLANK$&":"
150 CONDENSED_ON$=CHR$(15)
160 CONDENSED_OFF$=CHR$(18)
170 ENLARGED_ON$=CHR$(14)
180 ENLARGED_OFF$=CHR$(20)
190 FORM_FEED$=CHR$(12)
200 PRINTER$="PIO"
210 MYNAME$="Curtis Alan Pro
vance"
220 ADDRESS$="17 Constance S
treet "
230 CITY_STATE$="Merrimack,
NH  03054"
240 PHONE$="(603) 424-7624"
250 CALL CLEAR
260 DISPLAY "The program gen
erates coversfor cassette ca
ses.  You mayenter names 48
characters"
270 DISPLAY "long, tape coun
ter data, andtime informatio
n (both 7    characters long
).  You are"
280 DISPLAY "limited to ente
ring 14 namesper side.  If y
ou wish to   enter a blank l
ine or comma,"
290 DISPLAY "include quotes
- "" "".  Don'tworry if you
make a mistake,you will be a
ble to edit all"
300 DISPLAY "your entries be
fore they areprinted.":::::"
(PRESS ANY KEY TO CONTINUE)"
310 CALL KEY(0,K,S)
320 IF S=0 THEN 310
330 CALL CLEAR
340 TITLE$=""
350 FIRST_TIME=1
360 NO_COUNTER=1
370 NO_TIME=1
380 FOR X=0 TO 1
390 FOR Y=1 TO 14
400 NAME$(X,Y)=""
410 COUNT$(X,Y)="       "
420 TIME$(X,Y)="       "
430 NEXT Y
440 NEXT X
450 DISPLAY "Do you want to
enter tape   counter informa
tion?"::
460 CALL KEY(3,K,S)
470 ON 1-(K=89)-2*(K=78)GOTO
 460,480,490
480 NO_COUNTER=0
490 DISPLAY "Do you want to
enter time   information?"::
500 CALL KEY(3,K,S)
510 ON 1-(K=89)-2*(K=78)GOTO
 500,520,530
520 NO_TIME=0
530 DISPLAY :"Please enter/a
ccept title:  <------- 28 le
tters ------->":
540 DISPLAY TITLE$:
550 INPUT "":TEMP$
560 IF TEMP$="" THEN 580
570 TITLE$=TEMP$
580 TITLE$=SEG$(TITLE$&SEG$(
BLANK$,1,31-LEN(TITLE$)),1,3
2)
590 SIDE1=2
600 CALL CLEAR
610 X=2-SIDE1
620 DISPLAY "Please enter/ac
cept names ofitems (only the
 first 48    characters will
 be printed).":"SIDE";1+X:
630 FOR Y=1 TO 14
640 DISPLAY NAME$(X,Y):STR$(
Y);" ";
650 INPUT "":TEMP$
660 IF TEMP$="" THEN 680
670 NAME$(X,Y)=TEMP$
680 IF NAME$(X,Y)="" THEN 71
0
690 NAME$(X,Y)=SEG$(NAME$(X,
Y)&BLANK$,1,48)
700 NEXT Y
710 TOTAL(X)=Y-1
720 IF NAME$(X,1)="" THEN 93
0
730 IF NO_COUNTER THEN 830
740 CALL CLEAR
750 DISPLAY "For each title,
 input/accepta tape count (7
 characters  maximum)."
760 FOR Y=1 TO TOTAL(X)
770 DISPLAY :NAME$(X,Y):COUN
T$(X,Y):
780 INPUT "":TEMP$
790 IF TEMP$="" THEN 820
800 TEMP$=TEMP$&BLANK$
810 COUNT$(X,Y)=SEG$(TEMP$,1
,7)
820 NEXT Y
830 IF NO_TIME THEN 930
840 CALL CLEAR
850 DISPLAY "For each title,
 input/accepta time mark (7
characters   maximum)."
860 FOR Y=1 TO TOTAL(X)
870 DISPLAY :NAME$(X,Y):TIME
$(X,Y):
880 INPUT "":TEMP$
890 IF TEMP$="" THEN 920
900 TEMP$=BLANK$&TEMP$
910 TIME$(X,Y)=SEG$(TEMP$,LE
N(TEMP$)-6,7)
920 NEXT Y
930 SIDE1=SIDE1-1
940 IF SIDE1 THEN 600
950 DISPLAY "Would you like
to edit your information at
this time?"::
960 SIDE1=2
970 CALL KEY(3,K,S)
980 ON 1-(K=89)-2*(K=78)GOTO
 970,990,1010
990 FIRST_TIME=0
1000 GOTO 530
1010 OPEN #1:PRINTER$
1020 PRINT #1:CONDENSED_ON$;
EDGE$
1030 FOR X=1 TO 0 STEP -1
1040 TOTAL(X)=-TOTAL(X)*(NAM
E$(X,1)<>"")
1050 FOR Y=1 TO TOTAL(X)
1060 PRINT #1:": ";NAME$(X,Y
);COUNT$(X,Y);TIME$(X,Y);" :
"
1070 NEXT Y
1080 FOR Y=Y TO 14
1090 PRINT #1:BORDER$
1100 NEXT Y
1110 PRINT #1:EDGE$
1120 NEXT X
1130 PRINT #1:": ";ENLARGED_
ON$;TITLE$;ENLARGED_OFF$:BOR
DER$:EDGE$
1140 PRINT #1:CENTER$(MYNAME
$):CENTER$(ADDRESS$):CENTER$
(CITY_STATE$):CENTER$(PHONE$
):BORDER$:EDGE$:FORM_FEED$
1150 PRINT #1:CONDENSED_OFF$
1160 CLOSE #1
1170 DISPLAY "Want to make a
nother cover?"::
1180 CALL KEY(3,K,S)
1190 ON 1-(K=89)-2*(K=78)GOT
O 1180,330,1200
1200 CALL CLEAR
1210 STOP
```



T R I A L W A R E   b y   M A R T Y   K R O L L   J R

COOTER BUG ($5.00) - runs in console basic - speech optional with TE2 and speech synthesizer. Disk or cassette.

Cooter Bug is a non-violent game that young children can enjoy. For 1 to 4 players. It is a very colorful game; simple to play, yet very appealing. The objective of the game is to build your Cooter Bug, which consists of one body, one head, two eyes, two antennas, one mouth, and six legs. The first person to build a complete cooter wins the game.

GEOMETRIC SHAPES ($5.00) - requires Extended Basic Cartridge. Disk or Cassette.

This program graphically illustrates many geometric shapes, labels the dimensions, displays corresponding formulas, and calculates volume, perimeter, area or circumference. It is an enjoyable educational program.

Some of the geometric shapes you may select include square, rectangle, triangle, trapezoid, circle, sphere, cylinder, cone, and rectangular pyramid.

MORSE CODE TUTOR ($10.00) - requires Extended Basic Cartridge and the 32K memory expansion (internal or external). Disk or cassette

Morse Code Tutor is for both beginners and advanced Morse Code users. It enables the beginner to learn the Morse Code system. It allows both the novice and advanced Morse Code user to improve his code recognition and speed.

This program allows you learn the code, select the speed of transmission, select the pitch for transmission, and test your ability to recognize transmitted code. In addition, the machine language subprogram incorporates a routine that can initiate actual code transmission over the air.

DISASSEMBLER ($10.00) - Requires disk system, 32K memory expansion, and either Editor/Assembler or Mini Memory module.

With this program you have the choice of these 3 disassembly formats: Mnemonic instructions, data or text. Outputs to screen, printer or disk. Contains an option to output source code which is completely compatible with the editor of the Editor/Assembler. The commented source code for this program is included on the disk.

CATLOGING LIBRARY ($10.00) - Requires disk system, 32K memory expansion, and either Editor/Assembler or mini Memory module.

This disk cataloging program is capable of cataloging 900 files and 123 disks on each set of data files. One DSSD disk holds more than 5 sets of saves files, enabling storage of more that 5000 programs and 600 disks on one data disk. Reload data for later updates or printouts. Print entire catalog either file by file, disk by disk, or selectively by disk. All printouts either in single or multiple columns. Much, much more!!!

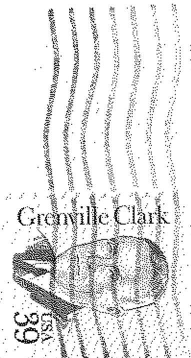
****

To obtain any of the above, ask a friend, or send $3.50 (for disk & postage, for each program, to:
Marty Kroll Jr.   218 Kaplan Avenue   Pittsburgh, Pa   15227

Try the programs. If you like them, please send the author the requested contribution listed above for each program. Thank you.

WEST PENN USERS GROUP
% John F. Willforth
R.D.#1 BOX 73A
JEANNETTE, PA
15644

PITTSBURGH PA
MAY 12
PM
1985



MICKY SCHMITT
196 BROADWAY AVE.
LOWER BURRELL PA 15068