ROCKY MOUNTAIN 99'ers

TIC TALK

VOL III, NO 6 DENVER, COLORADO USA FEBRUARY 1984

GOOD NEWS

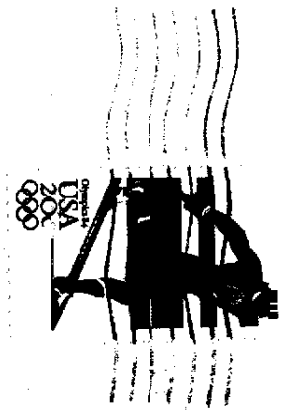
I made mention in the past that software would flourish, now that TI has stepped aside. Look at TIC's Add this month and see all the new titles that are available. I know, most of those are distributed by TI, but I can't remember when so many new titles have ever been introduced in such a short time frame. We are getting wind of many more by the Third Party Companies. It has started and I feel will continue.

How about the news from the California company by the name of CORCOM²! They have a 32k Memory Card and an RS232 Card available now. They will have a Peripheral Expansion Box in a month or two. They are working on a new Computer nicknamed the PHOENIX that looks real exiting. Designed to use TI software plus much much more with the capabilities of expanding into a very good Business Computer. Watch for further articles when they are made available to us. We have a direct communication link and will be awaiting further word.

FEBRUARY MEETING CHANGED!

We will meet on the 7th of February, which is a change to the first Tuesday. There is a big Valentine Dance on the 14th so our meeting had to be changed. Mark your calanders as such. The rest of the year will remain the 2nd Tuesday.

* * ROCKY MOUNTAIN 99ers * *
P.O. Box 3400
Littleton, CO 80161



Central Alabama
99/4A Users' Group
551 Larkwood Dr.
Montgomery AL 36109

* Do you see stars on the label *
* this means your membership is *
* now due. Send in your renew- *
* al today so you don't miss a *
* single issue of TIC-TALK!!! *

TI-WRITER

By Wayne R. Luedtka
344-5140

Don't go and  over a little  blow your brains out thing like the graphics mode.

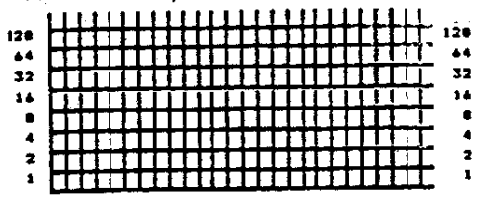
Last month I told you, you could do bit graphs with TI-WRITER. Well this month I'll tell you how to do it. First a few words of caution, remember that TI-WRITER will always put a carrige return (CHR$(13)) , and a Line feed (CHR$(10)) unless you tell it other wise. We do this by putting a (.CR) or (.LF) after the output device , eg. PIO.LF . This tells the computer not to output that code at the end of the line. With the Gemini I found that I have to leave it in the mode in which the printer adds a line feed after the Carriage return is sent. Other-wise the computer will do wierd things. For instance with the 8 in the transliterate command " .TL 43:27,75,5,0,128,96,8,63,0" which makes up the character " ", if printed with the command "PIO.LF" the computer will see the 8 as some thing other than the correct character you wanted.

For example in the last months news letter, the characters after the ASCII codes 4,13,20 , should have been: , , respectively . When I printed out the letter, the printer put in that weird character, then spaced over half a page and printed the CHR$(63) which turned out to be the "?".

The second thing you might keep in mind is that the computer and the printer have less intelligence than a common earth worm . After all the computer won't do anything unless you tell it; what ,when ,where ,and most important HOW! . Next ,read through chapter 6 on THE FUNDAMENTEALS OF DOT MATRIX PRINTING in the Gemini book. The next thing you need is some graph paper for your preliminary drawing . What-you say you don't have any graph paper ? Well, make some. How? Well you have one of the most powerfull graph paper making tools right at your finger tips. With TI-WRITER, try (.TL 1:27,76,10,0,255,255,3,3,3,3,3,3,3,3).
This will make the simbol "L". By putting a lot of these together, you can form one of these "LLLLLLLL". Next trick is to form a bunch of boxes out of your new found object. To do this , we must put the printer in a mode so that there are no spaces between the lines. All that is required for this is a line spacing of 15/144 of an inch. To do this, use the CTRL U, then ASCII 27, (FCTN R), 3, ASCII 15, (SHIFT O). Remember, change the spacing back to 1/6 inch when youre done. This can be found on page 128 of the manual.

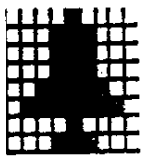
(Cont. from page 2)

With TI-WRITER you should be able to make a box which looks like this.



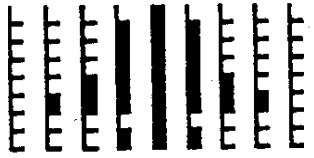
The numbers on the side represent the values for the dots in that line. All the dots in each coluan must be added up to get the firing order of the pins in the print head. Now take the set of numbers you have compiled, and add the commands for the printer and transliterate command ".TL 5:27,75,??,0,". The "??" are for the number of dot codes to follow. The "0" tells the printer that you plan to send it less than 255 dot codes. When using TI-WRITER, you can only get one line of dot codes with the .TL "Transliterate" command.

Let's try and make a rocket. Draw it out on your graph paper first.



Then convert each coluan into dot codes,

0 4 12 125 255 125 12 4 0



and add the Transliterate command for a finial result of
".TL 5:27,75,9,0,0,4,12,125,255,125,12,4,0"

Till next month 

<<<<< CLASSIFIED ADDS >>>>>

FOR SALE - CC-40 with Finance Moduled $165.00 contact Bruce Pattison - 733-5701

WANTED - Dot Matrix Printer contact Brian Wrenshall - 935-3048

FROM THE ASSEMBLY LINE

by Mike Holmes

This is the first of a series of articles by the members of the assembly language S.I.G. In this first article I was planning to discuss the equipment necessary to write programs for the 99/4A in assembly language. I decided, however, that with the uncertainty of finding some of the required equipment it would be better to spend this space examining the uses of assembly language. The assembly language is not the most direct link between people and computers but it is easier to use than the binary machine language that computers can understand directly. Assembly is a low level language. This means that it is not very close to human language. This has the disadvantage that assembly is not an easy language to learn, and the advantage that in assembly a programer can do things that are not possible in the high level languages like BASIC or PASCAL. In fact most of the high level languages are built up of command in assembler. These commands are broken down into machine language either by an interpreter (as in TI Basic) or a compiler. The compilation of a program is slower than interpreting initially but results in fast running machine code (also called object code). In contrast interpreted languages are reinterpreted every time a program is run This results in slower programs. Both compiled and interpreted languages are limited to the commands built into the language. This limits the flexiblility of the language to some extent. The assembler on the other hand is only limited by the user's understanding of the computer system he is working on.

When you first pick up the TI Editor Assembler Manual you are overwhelmed by the sheer volume of information that t contains. The fact that all of this information seems to have very little logic to it's organization does not help either. The main problem is that there are very few examples of assembler programs in the manual. Those that are there are poorly documented and hard to follow. In my case confusion persisted for quite a while. then I found an article in the 99'er entitled "Magic Crayon". The article itself did not impress me to any great extent, but the program!! Ah that program had comments in it even better the comments made sense! Since the program is copyrighted I cannot reproduce it here, but if you are interested in assembly language I highly recommend that you try to get a hold of a copy.

After reading entering and assembling this program several times trying to get it to work correctly I found that I had actually found out how to do something with assembly language. Now it was time for me to go on and attempt to write my own program. First I needed an idea. I found it one day when I was trying to edit a file on the editor assembler. I couldn't remember the name of the file. I had to quit and run the basic catalog program I put on every disk in order to find out the name of the file I wanted to edit. I decided to write an assembly language disk catalog program.

(Continued from page 4)

I did not, at the time realize that this would be as difficult as it turned out to be. The problem is not that assembly language is difficult but that the doccumentation for it is aimed at those with some assembly language batground. There is no good text covering TMS 9900 assembler. My only advice is that all the information that you need is in the book somewhere?! The trick is in finding that bit of infomation that you actually need. For example in my disk catalog progam I went round and round for several weeks trying to get a screen display before I realized that in setting up text mode I had turned off the bit that allows screen displays. A little later I found that the keyboard scan routine sets the condition bit in the GPL status byte when a key is pressed. The problem is that this bit is also set when you try to access a non existant device. This means that the GPL status byte must be cleared every time the routine DSRLNK is used.

The following is the source code for my first version of the Assembly language disk cataloger. I hope that the comments I have included will be of some help to those of you who are starting out. There are still several bugs in this version that I haven't ironed out. First, when the prompt is displayed if you do not enter a valid drive number you lock up the system and have to turn it off and back on again. Second, you must reload the program for every disk that you wish to catalog. Third, file sizes and types are not displayed. This is because I have not yet figured out how to handle internal format numbers. Finally when the program returns to the calling screen it changes the foreground color of the screen to cyan(? I think). This condition continues until a new selection is made from the E/A entry screen. I am presently working on a program which may fix all of these problems. I also hope to be able to tell you if you try to access a drive which is not attached. When this new version is complete I will send it to the newsletter for publication.

```
* ASSEMBLY LANGUAGE DISK CATALOGER
* BY MIKE HOLMES
* ROCKY MOUNTAIN 99'ERS
* ASSEMBLY LANGUAGE S.I.G.
*
DEF CAT
REF VSBW,VMBW,VMBR,VSBR Ref statements refer to external
REF DSRLNK,KSCAN,GPLLNK,VWTR subroutines which are predefined
*
* DEFINE CONSTANTS AND VARIABLES
*
PABBUF EQU >1920 This is the address of a buffer in CPU RAM
PAB EQU >1900 This is the address in VDP RAM for the P.A.B.
SCANO EQU >8ADA The number of the keyboard to scan is put here.
```

(Continued from page 5)

```
KCODE EQU >8375 The Ascii number of the key which was pressed is found
STATUS EQU >837C The address of the grom status byte.
PNTR EQU >8356 Pointer to the device name length in the PAB.
SAVRTN DATA 0 Save one word here for the return address.
ST DATA 0 Save one word here for status from PAB.
PDATA DATA >0000,PABBUF,>0000,>0000 This is the PAB.
DATA >0005 This is the length of the device name (Byte 9 of PAB).
TEXT 'DSK1.' Description of the device is part of the PAB.
EVEN This advarces assembly to an even word boundary.
READ BYTE >02 PAB OP Code to read record.
CLOSE BYTE >01 PAB OP Code to close file
STAT BYTE >09 PAB OP Code to get status from PAB.
EVEN Advance to even word boundary.
NAME BSS >0A Ten bytes to store names of files and disks.
MREGS BSS >20 User defined workspace registers, (32 Bytes).
REQ1 TEXT 'ENTER MASTER DISK: 1' Text for prompt.
EVEN Advance to even word boundary.
ZERO BYTE >00 One byte with a constant value of Zero.
BLANK BYTE >20 Ascii code for a blank character (decimal 32).
BUFFER BSS 80 80 byte CPU memory buffer to accept records read.
MASK1 DATA >0001 Represents the condition at end of file.
* BEGIN PROGRAM
*
CAT MOV R11,@SAVRTN Save return address to calling scrreen.
LI R0,>F001 Prepare to set up Text mode in VDP R1.
MOVB R0,@>83D4 This address maintains the screen time out parameters.  SWPB R0
Make R0 >01F0 to set up text mode >01 = register.
BLWP @VWTR Writes a value of >F0 to VDP R1
LI R0,>0715 Prepare to write colors black on blue to VDP R7.
BLWP @VWTR Write >15 (black on light blue) to VDP R7.
*
* CLEAR NAME ARRAY
*
BL @CLNAME Call subroutine to make 10 bytes of NAME blanks
*
* MOVE PAB TO VDP RAM
*
LI R4,PAB VDP RAM Location to take PAB.
LI R1,PDATA Beginning of PAB Description in CPU RAM,
LI R2,>20 32 bytes to write to VDP starting at PAB.
BLWP @VMBW Write PAB to VDP RAM.
*
* REQUEST DRIVE TO CATALOG
*
DRIVE CLR R10 Clear R10 for error check routine.
BL @CLEAR Call clear screen subrcutine.
LI R4,450 Point on screen to start writing text.
```


(Continued from page 6)

```
LI R1,REQ1 Address of prompt to be written to the screen.
LI R2,>0014 20 Bytes to write to screen.
BLWP @VMBW Write prompt on the screen.
*
* SELECT DRIVE TO READ FROM
*
SELEC MOVB @ZERO,@SCAN0 Select keyboard to scan.
RS BLWP @KSCAN Scan keyboard Number zero.
MOVB @STATUS,R0 See if key was pressed.
JEQ RS Try again if no key was pressed.
MOVB @KCODE,R9 Move ASCII code of key returned to R9.
SWPB R9 Make code least significant byte.
CI R9,>0D If enter key was press use default drive.
JEQ DEFLT Branch to default handling.
BL @ERCK Call error checking routine.
CI R10,>0001 Check for errors.
JEQ DRIVE If error is found request another drive.
LI R0,469 Address of drive number in screen prompt.
MOV R9,R1 Move drive number to R1 to add to prompt.
SWPB R1 Make drive number MSB of R1.
BLWP @VSBW Write drive number in prompt.
LI R7,>FFF Delay before continuing.
LI DEC R7 Decreament delay register by one.
JNE LI If R1 is not zero then decreament R1 again.
*
LI R0,PAB+13 Location of drive number in VDP PAB.
BLWP @VSBW Insert drive number into VDP PAB.
*
* OPEN FILE FOR INPUT
*
DEFLT LI R6,PAB+9 Set pointer to device name length in the PAB.
MOV R6,@PNTR Move to pointer address.
MOVB @ZERO,@STATUS Clear status byte.
BLWP @DSRLNK Open file with DSRLNK subroutine.
DATA 8 Required data for DSRLNK.
*
* READ RECORD
*
CLR R7 Clear file counter register.
BL @CLEAR Call clear screen subroutine.
NEXTRD LI R0,PAB+8 Prepare to read status byte from PAB.
LI R1,ST Load CPU address for status byte duplicate.
BLWP @VSBR Read PAB status byte.
MOVB @ST,R12 Transfer to R12.
COC @MASK1,R12 Check for end of file condition.
JNE CON If not EOF then continue.
```

(Continued from page 7)

```
JMP CLOSIT If EOF Tthen call close file routine.
CON MOVB @READ,R1 Load read opcode (clears errors for each read).
LI R0,PAB Set address to beginning of PAB in VDP RAM.
BLWP @VSBW Write OP CODE into PAB.
MOV R6,@PNTR Reset pointer to device name length.
MOVB @ZERO,@STATUS Clear GPL status byte.
BLWP @DSRLNK Read one record.
DATA 8
*
* WRITE TO CPU BUFFER
*
LI R0,PABBUF Load address of VDP buffer.
LI R1,BUFFER Load address of CPU buffer.
LI R2,80 Move 80 bytes from VDP to CPU memory.
BLWP @VMBR Read 80 bytes from VDP to CPU.
*
* FIND NAME OF DISK OR FILE
*
CI R7,>0000 Check to see if this is the first value read.
JNE SETNME If not first file then start interpreting name.
CLR R3 Clear R3 to hold NAME array address.
LI R13,3 Set R13 to the third colum in the first line.
SETNME CLR R5 Clear R5 NAME length counter.
MOV  R13,R8  R13  contains present screen write location for writes LI R3,NAME
Load Address of the NAME array.
LI R4,BUFFER Load Address of the CPU RAM buffer.
NEXT MOVB *R4,R8 Move the byte at the address in R4 to R8.
CI R8,>24 Check R8 for lowest allowable ASCII code.
JLE CONT If not legal code then continue.
CI R1,>5A Check R8 for highest allowable ASCII code.
JGT CONT If not legal then continue.
MOVB R8,*R3 Move legal code into name array.
INC R3 Increment NAME array one byte.
INC R5 Increment R5 NAME length counter one byte.
INC R4 Increment R4.
JMP NEXT Read another letter.
*
* PRINT NAME FROM DISK TO SCREEN
*
CONT LI R1,NAME Load address of NAME in R1.
CI R5,>0000 See if R5 (character count) is zero.
JNE C1 If not then continue from label C1.
BL @NEXTSC If no valid characters then call NEXTSC.
JMP CLOSIT Then goto close file routine.
C1 DECT R5 Subtract two from the total number of chars.
MOV R5,R2 Move number of characters in NAME to R2.
BLWP @VMBW Write file or disk name to screen.
```


(Continued from page 8)

```
CI R7,>0000 Is this the first NAME read?
JNE SK If not then go to label SK.
AI R13,40 Add 40 to R13 to advance screen position to next line SK CI R7,22 Is
this the 22nd NAME read?
JNE CEOF If not go to label CEOF.
LI R13,60 Otherwise set R13 to 60 to change columns for print.
CEOF AI R13,40 Add 40 to R13 to skip another line.
INC R7 Add one to the number of NAMEs read in R7.
CI R7,46 Have 46 NAMEs been read?
JNE SK2 If not then go to label SK2.
BL @NEXTSC Otherwise call routine NEXTSC.
SK2 BL @CLNAME Call clear NAME subroutine.
LI R0,PAB+8 Prepare to clear status byte from PAB.
CLR R1 Clear R1.
BLWP @VSBW Write zero to PAB status byte.
JMP NEXTRD Go to label NEXTRD.
*
* CLOSE FILE
*
CLOSIT MOVB @CLOSE,R1 Load close OP CODE.
LI R0,PAB Load beginning of PAB in VDP RAM.
BLWP @VSBW Write close OP CODE into PAB.
MOV R6,@PNTR Reset pointer to device name length.
MOVB @ZERO,@STATUS Clear GPL status byte.
BLWP @DSRLNK Close the file.
DATA 8 Required data for DSRLNK.
MOV @SAVRTN,R11 Reset return address to calling screen.
CLR R1 Clear R1.
MOVB @ZERO,@STATUS Clear GPL status byte.
B *R11 Return to the calling screen.
*
* SUBROUTINES
*
*
* CLEAR SCREEN
*
CLEAR LI R3,960 There are 960 bytes on the screen.
CLR R0 Clear R0 to write to the first screen location.
MOVB @BLANK,R1 Move a blank byte to R1.
CLR1 BLWP @VSBW Write a blank to the screen.
INC R0 Add one to the screen address.
DEC R3 Subtract one from counter (R3).
JNE CLR1 If counter is not zero then write another byte.
RT Return to calling routine.
*
* ERROR CHECK
*
```

(Continued from page 9)

```
ERCK CI R9,>0031 Compare ASCII code for byte in R9 to code for 1.
JLT ETONE If R9 is less than 1 then this is an error.
CI R9,>0033 Compare ASCII code for byte in R9 to code for 3.
JGT ETONE If R9 is greater than 3 the this is an error.
JMP RET Otherwise return to calling routine.
ETONE LI R10,>0001 Set R10 to 1 to indicate an error.
RET NOP Do nothing for one cycle.
RT Return to calling Routine.
*
* CLEAR NAME ARRAY
*
CLNAME LI R5,NAME Load address of NAME array in R5.
LI R3,>09 Load 9 into counter (R3).
CLN1  MOVB  @BLANK,*R5+  Move blank byte in array and increment address in R5.
DEC R3 Subtract one from counter.
JNE CLN1 If counter (R3) is zero then go move another.  MOVB  @BLANK,*R5 Move
one more byte into array.
RT Return to calling routine.
*
NEXTSC BLWP @KSCAN Scan for key press.
MOVB @STATUS,R0 See if key was pressed.
JEQ NEXTSC If not then wait until one is.
LI R13,83 Load 83 into R13.
LI  R2,940  940 is  one  row less than the full screen image table.  LI R0,41
Start writing at first character of second row.
MOVB @BLANK,R1 Put a blank byte in R1.
CLS1 BLWP @VSBW Write a blank to the screen.
INC R0 Add one to screen offset.
DEC R2 Subtract 1 from R2.
JNE CLS1 If R2 is not zero then do it again.
RT Return to the calling routine.


END CAT End of program with label to run automatically.
```

I really hope that this program will help you as you try to understand the assembly language of the TI 99/4 family of computers. If you have any luck with your programs those of us in the assembly language Special Interest Group would like to see what you are doing. If you are having trouble we would like to see if we can help. If you find the assembler of interest please feel free to call me or talk to me or anyone else in the group at the next meeting. Until then good luck and good programing.

MAIN MENU ?

Martha Meeg

In the november issue of TIC TALK, there was a menu program for X-BASIC and a disk drive , from the Tri-State User' Group. Using a program named 'LOAD", it provides a menu of X-BASIC programs automatically when X-BASIC is selected.

This gets you into any of the menu programs,but does not provide for returning to the main menu when you are done with the desired program.

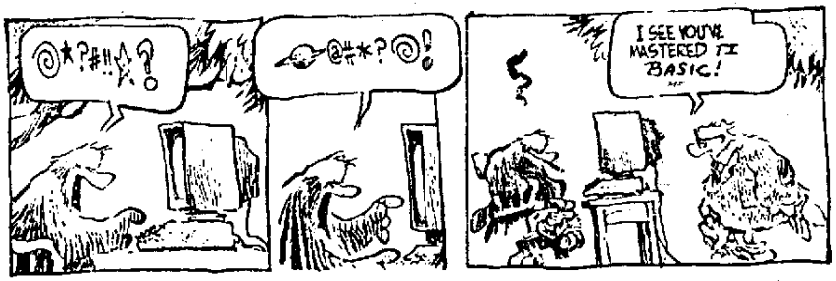
Below is a listing of the lines required to add this capability.These lines must be added to each of the menu programs. They should ge placed at the point in the program at which it asks if you want to play (or run) it again . They should be after the point at which a "Y", returns control to the beginning of the program. A "N" response (or any key not "Y", in some programs) should transfer control to the first line of the subprogram below (instead of to the "END" line).

It uses the CALL KEY statement. The ASCII code for "Y" is 89 , and for "n" is 78.

You will have to change the line numgers used to fit each program. If the program doed not have "room" for the addition , you can use the RESEQUENCE function to open up the line numgers in your puogram.

Also change line 10 to display the "RETURN TO MAIN MENU?" prompt at a position compatible with the final screen format of yur program.

```
10 DISPLAY AT(24,5):"RETURN TO MAIN MENU?(Y/N)"
20 CALL KEY(0,KEY,STATUS)
30 IF KEY=78 THEN 60
40 IF KEY<>89 THEN 20
50 RUN "DSK1.LOAD"
60 END
```



<<<<< CLASSIFIED ADD RATES >>>>>

MEMBERS - FREE (25 word max) We must have you add by the 20th of the month to assure insertion in the next issue. Call 979-6677 or mail to BOX 3400 Littleton, CO 80161. NON-MEMBERS not allowed!

<<<<< DISPLAY ADDS >>>>>

RATES: 10 in X 7.5 in - $30.00
4.5 in X 7.5 in - $16.00
4.5 in X 3.5 in - $9.00

ALL DISPLAY ADDS must be camera ready and must be received before the 20th of the of the month and accompanied by a check made out to the ROCKY MOUNTAIN 99ers. Since the Club is a non-profit organization all money collected for advertising goes toward publishing costs of this newsletter.

~~~~OFFICERS and CHAIRMEN~~~~

PRESIDENT	RON KUSESKI	444-1797	** PROGRAM	TED MICHELSEN	986-3513
VICE-PRES.	TED MICHELSEN	986-3513	** LIBRARIAN	PETE CROWELL	750-5949
SECRETARY	MARTHA WEEG	455-4389	** MEMBERSHIP	MARTHA WEEG	455-4389
TREASURER	KEN MONSON	233-1788	** ASSEMBLY SIG	MIKE HOLMES	751-7945
EDITOR	LLOYD MAPLE	979-6677	** What's your SPECIAL INTEREST		750-5949

EDITOR/ASSEMBLER SIG

Normally, the E/A Group meets the first Tuesday of the month. However since the Club meeting is scheduled this month on the first Tuesday, this will change for February. Check with Mike Holmes at February's main meeting for date, time, and place.

MULTIPLAN SIG

Anyone interested in learning more about the powerful program of MultiPlan and what it can do for you call BEN KRAMER 237-1056. He is very interested in starting a Special Interest Group on this Program.