ROCKY MOUNTAIN 99'ers

TIC TALK

VOL III, NO 3 DENVER, COLORADO USA NOV 1984
Non-member Subscription Rate - $7.50 Annually Single Copy Price - 75 cents

Hello out there!

My name is David Owen, and I think I am your new editor. The last time I said something similar, I was commandered to help edit, collate, staple, sticker and stamp a Submarine Veterans paper. Fortunately, this one is not a 2,000 mailer, and all I have to do is edit, and then send the mess to the printer and let him worry about it.

All articles and programs will be greatly appreciated (saves me a lot of time, besides I'm still new at this). Camera ready copy (or TI-Writer compatible disc) would be great. We'll be glad to receive anything though!!!!! All material should be in by the 15th of the month, so I can get this to you before the meeting date.

Don't forget to vote on the 6th, and I'll see you on the 13th!

NOVEMBER MEETING

NOVEMBER 13

Jefferson County Fairgrounds

Auditorium 7:00 PM

6th Ave. West to Indiana Ave.

by Mike Holmes

I know that it has been almost a year since I wrote part one of this article and I did promise to write a follow up when I had completed the program so here it is. This program was the first original program that I wrote in assembly language and I little knew when I started it that it would hold so many pitfalls. Some of these I mentioned in the first article but I wasn't through stumbling yet. When I wrote the last article I had completed the code in assembly language to read the disk catalog file and print the file names. The next item on the agenda was to retrieve the numerical information from the catalog and interpret that. My first step was to get out the disk controler manual and check out exactly what the information stored in the catalog meant. Fortunately TI is very free with this information, (unlike some other information that they guard with their very lives.) and this is very well documented in the manual. Unfortunately when TI stores any number on disk they use RADIX 100 floating point format. Fortunately you don't have to understand this stuff in order to use these numbers since there are several ROM routines which will convert numbers to and from floating point and so on. The routine I used in the program converts the Floating point number directly to a binary integer. The format of the numbers on disk consists of nine bytes as follows.

1. length of number in bytes - always 8 for floating point numbers.
2. exponent of 100
3. hexidecimal representation ot decimal number fills the remaining seven bytes.

The exponent and seven bytes of the number must be moved into the floating point accumulator in RAM before executing the BLWP @XMLLNK with a data statement to access the convert to integer routine in ROM. The integer number is then left in the first word of the Floating Point Accumulator for your use. O.K. now you have binary integer numbers representing the number of sectors on the disk, the number of used sectors on the disk, the sizes of files, the file type and the number of bytes in a record, how do you interpret this information? There are two routines which carry out this interpretation. If a number is to be displayed it must be converted to ASCII from binary. This is the routine to convert to displayed decimal ascii from binary.

1. The binary number is divided by 10 (decimal) and the remainder is kept.
2. The ascii value of decimal zero is added to the remainder to get the ascii code of the digit.
3. This digit is displayed at the right of the display field.
4. The quotent of the division is used to replace the original number and steps 1 on are repeated until the field is filled.

File types were determined through the use of a loop which counted down from the number stored as the file type. Each time through the loop the address of the string to be printed was updated by the length of the string so that when the file type reached 0 the address of the string to be displayed was all ready to go.

A couple of notes are in order. When you look at the text strings used for file descriptions you will see that I have called what are usually called program files M/I for Memory Image files. I feel that this is more descriptive of the file type but if you wish just change the TEXT statement to read ' PGM'. Make sure to leave the spaces before the file descriptions in order to get a readable display. If you are using a double sided drive or double density controler you should be able to display most of the files on a disk on one screen. If your catalog runs over one screen the program halts and waits for you to press any key to see the files on the next screen.

Here are some ideas that you may want to try in order to adjust the program for your best use.

1. Make the program store the File information so that you can use the arrow keys to page back and forth between the pages of the catalog.

2. Fix the program to put the cursor in front of any program name on the screen and when you press enter have the program loaded and run. (This is a pretty tall order so don't undertake it unless you feel like going crazy.

```
************************************
*  ASSEMBLY LANGUAGE DISK CATALOGER*
*  VERSION 2 BY MIKE HOLMES        *
*  ROCKY MOUNTAIN 99'ERS           *
*  ASSEMBLY LANGUAGE S.I.G.        *
************************************
       DEF CAT
       REF VSBW,VMBW,VMBR,VSBR,GPLWS
       REF DSRLNK,SCAN,GPLLNK,VWTR
*------------------------------*
* DEFINE CONSTANTS AND VARIABLES *
*------------------------------*
PABBUF EQU  >1920    VDP Buffer for PAB
PAB    EQU  >1900    PAB location in VDP memory
SCANO  EQU  >83DA    Code for keyboard to scan
KCODE  EQU  >8375    Key Code returned by Keyboard scan
STATUS EQU  >837C    GPL Status byte
PNTR   EQU  >8356    Pointer to device name length in VDP
SAVRTN DATA 0        Save return address
ST     DATA 0        Saved status
NEST1  DATA 0       ¦Save nesting levels
NEST2  DATA 0       ¦
NEST3  DATA 0       ¦
NEXTS  DATA 0       ¦
D100   DATA 100      Decimal value of 100
PDATA  DATA >000D,PABBUF,>0000,>0000,>0005
       TEXT 'DSK1.'  These two lines make up the data for the PAB
       EVEN
KSAVE  BSS  >01      Save key press
READ   BYTE >02      PAB OP Code to read record
CLOSE  BYTE >01      PAB OP Code to close file
OPEN   BYTE >00      PAB OP Code to open file
Q      BYTE >51      Upper Case 'Q' for quit
Q1     BYTE >71      Lower Case 'q' for quit
D1     BYTE >31      ASCII value of '1' Lowest drive number
D4     BYTE >34      ASCII value of '4' Highest drive number
ENT    BYTE >0D      ASCII value for enter key
       EVEN
ETEXT  TEXT 'DRIVE #2 NOT FOUND'
MEN1   TEXT 'MENU'
MEN2   TEXT 'PRESS ''Q'' TO QUIT'
MEN3   TEXT 'PRESS ANY OTHER KEY TO'
MEN4   TEXT 'CATALOG ANOTHER DISK'
REQ1   TEXT 'ENTER MASTER DISK: 1'
       EVEN
ZERO   BYTE >00      Value of Zero
BLANK  BYTE >20      ASCII Code for blank character
MREGS  BSS  >20      Workspace for the catalog program
BUFFER BSS  80       80 byte buffer in memory for reads
```

```
*--------------------------------*
* BEGIN PROGRAM SET UP TEXT MODE *
*--------------------------------*
CAT     MOV  R11,@@SAVRTN  Save return address to Editor/Assembler
        LWPI MREGS         Load Catalog workspace
        BL @@CLEAR          Clear the entire screen
        LI   R0,>F001      Set up to write text mode parameters for timeout
        MOVB R0,@@>83D4     This value determins how screen will be restored
        SWPB R0            Prepare to set up text mode in VDP R1
        BLWP @@VWTR         Write to VDP R1
        LI   R0,>0715      Set colors to black on lt blue VDP R7
        BLWP @@VWTR         Write colors to VDP R7
*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
* MAIN PROGRAM CALLS ALL SUBROUTINES *
*^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*
MAIN    LI R6,PAB+9        R6 Stores pointer to PAB name length
        BL @@GETDRV          Find Drive number to catalog and chedk validity
        BL @@GETNAM          Read and Display data from disk catalog
        BL @@CLEAR           Clear the screen
        BL @@CLOSOP          Close the catalog file
        BL @@MENU            Display the menu of options
        BL @@GETKEY          Wait for a key press
        CB @@KSAVE,@@Q        Check for capital 'Q'
        JEQ MAIN1          Quit if Q was pressed
        CB @@KSAVE,@@Q1       Check for lower case Q
        JNE MAIN           Start over if q was not pressed
MAIN1   LWPI GPLWS         Reload GPL work space for return to E/A
        BL   @@CLEAR         Clear the screen
        MOV @@SAVRTN,R11     Reset return address to E/A
        MOVB @@ZERO,@@STATUS Clear status byte
        RT                 Return to E/A
*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>*
*  ROUTINE TO GET DRIVE NUMBER TO CATALOG     *
*  CHECKS FOR VALID DRIVE NUMBER AND VALID PAB *
*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>*
GETDRV MOV  R11,@@NEST1     Save return address to main program
GD1     BL   @@CLEAR         Clear the screen
        BL   @@WRREQ         Write drive request to screen
        BL   @@GETKEY        Wait for keyscan and save results
        CB   @@KSAVE,@@ENT  Was enter key pressed?
        JEQ  GD2           Yes handle as a default
        CB   @@KSAVE,@@D1    Is the value returned less than 1?
        JLT  ERRORC        Yes display error message and try again
        CB   @@KSAVE,@@D4    Is the value greater than 4  (Cor-Comp card)
        JGT  ERRORC        Yes display error message and try again
        MOVB @@KSAVE,@@PDATA+13 Move drive number into correct PAB location
*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*
* INSERT DRIVE AS DEFAULT IN FUTURE ACCESSES *
*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*
        MOVB @@KSAVE,@@REQ1+19
*//////////////////////////////////////////*
* INSERT DRIVE INTO REQUEST STRING FOR FUTURE *
*//////////////////////////////////////////*
        BL   @@WRREQ         Rewrite the request with the proper drive number
GD2     LI   R0,PAB        VDP location for PAB
        LI   R1,PDATA      CPU location of data for PAB
        LI   R2,>0F        15 Bytes to write
        BLWP @@VMBW          Write into VDP memory
        BL   @@OPENOP        Open file for input (Default for enter key)
        MOVB @@STATUS,R0     Check status for non existent device
```

```
       JEQ   GD3            No error then continue
ERRORC BL    @@CLEAR         Clear the screen
       LI    R0,451         Write error message at this location on screen
       MOVB  @@KSAVE,@@ETEXT+7   Update error text
       LI    R1,ETEXT       Load error message address
       LI    R2,18          Length of error message
       BLWP  @@VMBW          Write to screen
       BL    @@GETKEY        Wait for a key press
       JMP   GD1            Restart routine
GD3    MOV   @@NEST1,R11    Reset return address to main program
       RT                   Return to main program
*****************************************************
*  ROUTINE TO SET PAB OP CODES FOR ACCESS TO CATALOG FILE *
*****************************************************
READOP CLR   R1             Clear R1
       LI    R0,PAB+8       PAB status byte address
       BLWP  @@VSBW          Clear PAB status byte for read
       MOVB  @@READ,R1       Load R1 with read OP code
       JMP   DSROP          Continue with DSR routine
OPENOP MOVB  @@OPEN,R1       Load Open OP code in R1
       JMP   DSROP          Continue with DSR routine
CLOSOP MOVB  @@CLOSE,R1      Load Close OP code in R1
DSROP  LI    R0,PAB         Load address of start of PAB
       BLWP  @@VSBW          Write selected OP code to PAB BYTE 1
       MOV   R6,@@PNTR       Set pointer to device name length in PAB
       MOVB  @@ZERO,@@STATUS   Clear status byte (if set DSR is nonexistent) ILE)
       BLWP  @@DSRLNK        Call DSR LINK Suproutine
       DATA  8              Required data
       RT                   Return to calling subroutine
*\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*
*  SUBROUTINE TO TRANSLATE NAMES FROM VDP BUFFER TO SCREEN *
*\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*
GETNAM MOV   R11,@@NEST1     Store Return address to main program
       BL    @@CLEAR         Clear the screen
       CLR   R7             Clear the file counter
       LI    R13,2          Set the screen location pointer
GN1    BL    @@READOP        Call subroutine to set PAB OP code to read
       LI    R0,PABBUF      Load address of VDP RAM buffer containing record
       LI    R1,BUFFER      Load address of CPU RAM buffer ro receive record
       LI    R2,80          Load number of bytes to transfer from VDP to CPU
       BLWP  @@VMBR          Transfer data to CPU buffer
*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*
*  SUBROUTINE TO GET NAMES OF DISKS AND FILES  *
*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*
NAMIT  CLR   R2             Clear pointer regester
       LI    R1,BUFFER      Load address of CPU buffer
       CB    *R1,@@ZERO      Compare Length byte to 0
       JLE   STOPIT         If length is zero or less stop
       INC   R1             Skip over name length byte in buffer
CONTIT CI    R7,>0000       Is this the first name?
       JEQ   WRIT           Yes write it out
       CI    R7,23          Is this the 23 name?
       JNE   SAMCOL         Yes write in the same column
       LI    R13,101        Otherwise write in second column 3rd row
SAMCOL CI    R7,46          Is this the 46th name?
       JNE   WRIT           No write it out
       BL    @@NEXTSC        Otherwise wait for a key press
       LI    R13,82         Start next name in first column of row 3
       LI    R7,>0001       Reset file count to 1
       JMP   GN1            Read another name
```

```
WRIT    MOV   R13,R0        Set screen location
        AI    R13,40        Add 40 to screen location pointer regester
        CI    R7,>0000      First name?
        JNE   WRIT2         No continue
        AI    R13,40        Yes skip one row for next name
*}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}*
* THIS ROUTINE SCANS CPU BUFFER FOR LENGTH BYTE OF THE FIRST FP NUMBER *
*}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}*
WRIT2   CB    @@BUFFER+1(R2),@@D8
        JEQ   WRIT1         If the byte value equals 8 the name has ended
        CI    R2,10         If the name length equals 10 this is the maximum
        JEQ   WRIT1         Continue
        INC   R2            Advance pointer
        JMP   WRIT2         Check next value
WRIT1   INC   R7            Increment file counter
        BLWP  @@VMBW         Write name on screen
        A     R2,R1         Add Length of name to start of buffer
        INC   R1            Increment buffer location
        BLWP  @@FLP          Branch to floating point routines
        JMP   GN1           Get another file name
STOPIT  BL    @@NEXTSC       Wait for key press
        MOV   @@NEST1,R11    Reset return address to main program
        RT                  Return to main program
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*
*  ROUTINE TO PREPARE TO SHOW ADDITIONAL FILE NAMES *
*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*
NEXTSC MOV   R11,@@NEST2    Save return to calling program
        BL    @@GETKEY       Wait for key press
        BL    @@PCLEAR       Clear part of screen
        MOV   @@NEST2,R11    Reset return address
        RT                  Return to calling program
*/////////////////*
* KEYSCAN ROUTINE *
*/////////////////*
GETKEY LWPI  GPLWS          Load GPL work space
        MOVB  @@ZERO,@@SCANO Scan keyboard zero
GK1     BL    @@SCAN         Branch to scan routine
        MOVB  @@STATUS,R1    Was a key pressed
        JEQ   GK1           No rescan
        MOVB  @@KCODE,@@KSAVE Save Keycode
        LWPI  MREGS         Reload catalog work space
        RT                  Return
*\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*
* ROUTINE TO WRITE MENU TO SCREEN *
*\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*
MENU    LI    R0,18
        LI    R1,MEN1
        LI    R2,4
        BLWP  @@VMBW
        LI    R0,210
        LI    R1,MEN2
        LI    R2,17
        BLWP  @@VMBW
        LI    R0,248
        LI    R1,MEN3
        LI    R2,22
        BLWP  @@VMBW
        LI    R0,289
        LI    R1,MEN4
        LI    R2,20
```

```
        BLWP  @@VMBW
        RT
*//////////////////////////*
* CLEAR SCREEN ROUTINES *
*//////////////////////////*
PCLEAR LI    R2,880      ~ Clears the screen starting at the first column of the
       LI    R0,81       ~ third row.
       JMP   CLR1
CLEAR  LI    R2,960      * Clears the entire screen
       CLR   R0          *
CLR1   MOVB  @@BLANK,R1   *
       BLWP  @@VSBW        *
       INC   R0          *
       DEC   R2          *
       JNE   CLR1        *
       RT                  Return to calling program
*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*
* ROUTINE TO WRITE DRIVE REQUEST TO SCREEN *
*&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&*
WRREQ  LI    R0,450
       LI    R1,REQ1
       LI    R2,>0014
       BLWP  @@VMBW
       RT
*-------------------------------------------------------*
* FLOATING POINT HANDLING ROUTINES FOR DISK CATALOGER *
* STATISTICS HANDLING                                   *
*-------------------------------------------------------*
       REF    XMLLNK      Ref xmllnk to gain access to the floating point
FLREGS BSS    >20         Workspace for floating point
FLDATA DATA   0
FLP    DATA FLREGS,GETSTS  Context switch vectors
*
TOTAL  DATA >0000          Store total sectors on disk (binary).
AVAIL  DATA >0000          Store available sectors on disk (binary).
USED   DATA >0000          Store used sectors on disk (binary).
*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>*
* MESSAGE STRINGS FOR DISK MESSAGES. *
*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>*
DSKMS1 TEXT 'TOTAL SECTORS = '
SIZE1  TEXT '     '
DSKMS2 TEXT 'AVAILABLE = '
SIZE2  TEXT '     '
DSKMS3 TEXT '   USED = '
SIZE3  TEXT '     '
*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*
* GET THE VALUES FROM THE CALLING ROUTINE'S WORKSPACE *
*<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*
GETSTS MOV   R13,R4        Copy the address of the calling workspace
       MOV   *R4,R0        Move value of screen address to floating point R0
       INCT  R4            Increment to the address of calling R1
       MOV   *R4,R1        Move buffer address value to fp R1
       INCT  R1            Increment to the address of calling R2
       AI    R4,12         Add offset to calling R7
       MOV   *R4,R7        Move value of file counter to fp R7
       CI    R7,>0001      Is this the disk name?
       JEQ   DISK          Yes, display the disk statistics.
       B     @@FILE         No, display the file statistics.
```

```
*****************************
* CODE FOR DISK STATISTICS *
*****************************
DISK    AI    R1,9          Add 9 to the buffer offset (skip the file type)
        BL    @@MOVFAC       Move 8 bytes into the floating point accumulator
        BL    @@CNVINT       Convert floating point number in FAC to integer.
        MOV   @@FAC,@@TOTAL  Move integer value to total sectors.
        MOV   @@TOTAL,@@USED Copy total into used sectors
        BL    @@MOVFAC       Move another 8 bytes into FAC
        BL    @@CNVINT       Convert to integer
        MOV   @@FAC,@@AVAIL  Move value to available sectors
        S     @@AVAIL,@@USED Subtract avail from the copy of total to get used
        LI    R0,19          Load 19 as screen address in R0
        LI    R1,DSKMS1      Message to write to screen
        LI    R2,20          Write 20 characters
        BLWP  @@VMBW          Go ahead.
        LI    R3,35          Load 35 in R3 (beginning of display field).
        LI    R0,38          Load 38 in R0 (end of display field).
        MOV   @@TOTAL,R5     Move total to R5 (Binary number to display).
        BL    @@CNVASC       Convert to ascii display and strip leading 0's
        LI    R0,42          Load location to display next message.
        LI    R1,DSKMS2      Load start of message
        LI    R2,29          Write 29 bytes in message
        BLWP  @@VMBW          Do it.
        LI    R0,57          End of next number field
        LI    R3,54          Beginning of next number field
        MOV   @@AVAIL,R5     Number to convert
        BL    @@CNVASC       Convert to ascii
        LI    R0,70          End of next number field
        LI    R3,67          Beginning of next number field
        MOV   @@USED,R5      Number to convert
        BL    @@CNVASC       Convert to ascii
        JMP   GOBAK          Return to calling program
*****************************
* CODE FOR FILE STATISTICS *
*****************************
TYPE    DATA >0000           Store file type (binary).
SIZE    DATA >0000           Store file size (binary).
BPR     DATA >0000           Store bytes per record (binary) (not used)
TYPES   TEXT ' D/F'     *    Display form of file types
        TEXT ' D/V'     *
        TEXT ' I/F'     *
        TEXT ' I/V'     *
        TEXT ' M/I'     *
*
FILE    MOV   R1,R9          Copy buffer offset
        BL    @@MOVFAC       Move next 8 bytes to FAC
        BL    @@CNVINT       Convert to integer
        MOV   @@FAC,@@TYPE    Save as file type
        AI    R9,8           Add 8 to buffer offset
        MOV   R9,R1          Move new offset to R1
        BL    @@MOVFAC       Move next 8 bytes to FAC
        BL    @@CNVINT       Convert to integer
        MOV   @@FAC,@@SIZE    Save as file size
        AI    R9,8           Add 8 to buffer offset
        MOV   R9,R1          Move new offset to R1
        BL    @@MOVFAC       Move 8 more bytes to Fac
        BL    @@CNVINT       Convert to integer.
        MOV   @@FAC,@@BPR     Save as bytes per record
        AI    R0,11          Add 11 to the last screen location
```

```
        MOV   R0,R3          Copy to R3 (start of display field).
        INCT  R0             Add 2 to R0 (end of display field).
        MOV   @@SIZE,R5       Number to convert.
        ABS   R5             Take the absolute value of the number.
        BL    @@CNVASC        Convert to ascii and display
        LI    R4,TYPES       Start of array of display file types.
        MOV   @@TYPE,R5       Integer value of type
        ABS   R5             Take the absolute value of the file type.
        DEC   R5             Decrement type by 1
        JEQ   FIL1           If type is 0 then print file type from addr in R4
NXTTYP  AI    R4,4           Add 4 bytes to the file type address in R4
        DEC   R5             Subtract 1 from type
        JNE   NXTTYP         If type is not zero then do it again
FIL1    INC   R0             Add 1 to screen address in R0
        LI    R2,4           Put a length of 4 in R2
        MOV   R4,R1          Move address of display file type from R4 to R1
        BLWP  @@VMBW          Write to the screen
GOBAK   RTWP                 Return to calling routine.
*********************************
* CODE FOR CONVERSION TO INTEGER *
*********************************
VSPTR   EQU   >836E           Value Stack PointeR
FAC     EQU   >834A           FloAting Point accumulator
*
CNVINT  CLR   R12            Clear R12
        MOVB  R12,@@STATUS    Clear status byte
        LI    R12,>06FB      Initialize VSPTR to >06FB (reccommended by TI I
*                                                       don't know why.)
        MOV   R12,@@VSPTR     Move to VSPTR
        BLWP  @@XMLLNK        Call XMLLNK subroutine
        DATA  >1200          Call Convert to integer routine
        RT                   Return to calling routine.
*************************************************************
* ROUTINE TO LOAD FLOATING POINT ACCUMULATOR WITH 8 BYTES *
*************************************************************
D8      BYTE  8              This byte represents the length of a FP number.
        EVEN
MOVFAC  LI    R4,7           Set R4 to 7 bytes to move
        LI    R5,FAC         FAC is the address to move to
GETSTR  CB    *R1,@@D8        Check to see if this is the length of the number.
        JEQ   SKPLEN         Yes the continue
        DEC   R1             No back up one byte.
        JMP   GETSTR         Check again
SKPLEN  INC   R1             Skip over length byte.
ANOTH   MOVB  *R1+,*R5+      Move a byte from the buffer to FAC
        DEC   R4             Decrement the count of bytes to move.
        JNE   ANOTH          If R4 is not 0 move another byte
        MOVB  *R1,*R5        Move another byte to FAC (8 bytes total).
        INC   R1             Increment R1.
        RT                   Return to calling program.
*************************************************************
* ROUTINE TO CONVERT TO ASCII DISPLAY ON THE SCREEN AND *
* STRIP LEADING ZEROS.                                  *
* ASSUMES NUMBER IS IN R5 END OF FIELD TO DISPLAY       *
* NUMBER IS IN R0 AND THE BEGINNING IS IN R3            *
*************************************************************
TEN     DATA  >000A
A0      BYTE  48
        EVEN
*
```

```
CNVASC CLR  R4          Clear R4 to prepare for division.
       DIV  @@TEN,R4     Divide number in R4 and R5 (32 bits) by 10
       AI   R5,48       Add 48 (difference between binary 0 and ascii 0)
       SWPB R5          to the remainder in R5 and make MSB
       MOVB R5,R1       Move byte value of number to R1
       BLWP @@VSBW       Write number to the screen
       DEC  R0          Decrement R0 address
       MOV  R4,R5       Move dividend from R4 to R5
       C    R0,R3       Is R0 less than the beginning of the field?
       JHE  CNVASC      No continue conversion.
       INC  R0          Add 1 to address in R0 (beginning of field)
       AI   R3,2        Add 2 to address in R3 (end of field)
STRIP  BLWP @@VSBR       Read the byte at the address in R0
       CB   R1,@@A0      Is that byte an ascii 0?
       JH   CNX         If greater than 0 exit the routine.
       LI   R1,>2000    Load R1 with an ascii space.
       BLWP @@VSBW        Replace the ascii 0 with a space.
       INC  R0          Move to next screen address.
       C    R0,R3       End of the field?
       JLE  STRIP       No the strip another character
CNX    MOV  R3,R0       Reset R0 to the end of the field
       RT               Return to calling routine.
       END CAT          End of program to stop auto running remove CAT
*                       and this comment.
```

The Editor/Assembler SIG meeting will be Wednesday November 7, 1984 at Unique Systems, located at Bates and Broadway behind the Oak and Pine store.

<<<<< DISPLAY ADS >>>>>

RATES: 10 in X 7.5 in - $15.00
5.5 in X 7.5 in - $8.00
3 in X 7.5 in - $4.50

ALL DISPLAY ADDS must be camera ready and must be received before the 15th of the month and accompanied by a check made out to the ROCKY MOUNTAIN 99ers P.O. Box 3400, Littleton, CO 80161. Since the Club is a non-profit organization all money collected for advertizing goes toward the publishing costs of this newsletter.

<<<<< WANT AD RATES >>>>>

MEMBERS - FREE (25 word max) We must have your add by the 15th of the month to assure insertion in the next issue. Call 458-7315 or mail to BOX 3400 Littleton, CO 80161. NON-MEMBERS must use DISPLAY ADS!

COLUMBINE COMPUTERS

6926 W FREMONT PLACE
LITTLETON, CO 80123

795-5225 (Daytime) (Evenings) 979-6677

NOVEMBER 1, 1984 **HOLIDAY SPECIALS** NOVEMBER 1, 1984

$5.95 each 6 for $29.95 12 for $58.95

All titles below now in STOCK but prices good while supply lasts

EDUCATIONAL
Early Learning fun PHM3002
Equations PHM3100
Fractions PHM3095

INFORMATION MANAGEMENT
Home Financial Decisions PHM3006
Household Budget Management ..PHM3007
Personal Realestate PHM3022
Tax/Investment Record Keeping PHM3016

ENTERTAINMENT
A-Maze-ing PHM3030
Alpiner PHM3056
Attack PHM3031
Blasto PHM3032
Car Wars PHM3054
Hopper PHM3229
Hunt the WUMPUS PHM3023
Jawbreaker II PHM3194
Moonmine PHM3131
Munch Man PHM3057
Parsec PHM3112
Sneggit PHM3145
TI Invaders PHM3053
Tombstone City PHM3052

Many more Educational and Entertainment Module Software HEAVILY DISCOUNTED!! I have many modules in stock for the Holidays. You can pickup without waiting.

WHY NOT GIVE A 99/4A FOR A CHRISTMAS PRESENT!!!

99/4A CONSOLE w/Plastic Cover .....$ 79.00
99/4A CONSOLE W/TI SOFTWARE VARIETY PACK(4 cassette programs) .....$ 89.00

PERIPHERAL EXPANSION SYSTEM (Disk Dr, Controller, and 32k Memory) $399.95

PRINTER SPECIALS

EPSON RX-80 (100cps) .....$ 245.00
BMC BX/80 (80cps-sqr dot) .$ 269.00
POWERTYPE Daisywheel 18cps $ 385.00

MONITOR SPECIALS

BMC 13" COLOR w/Non-Glare.....$ 218.00
AMDEK 13" COLOR 1+ w/Non-Glare$ 259.00
COMREX 13" COLOR .....$ 279.00

INTRODUCTORY SPECIAL PRICES -- EPSON COMPUTERS -- INTRODUCTORY SPECIAL PRICES

PX-8 GENEVA Portable Computer 64k RAM CP/M .....$ 849.00

QX-10 256k RAM Standard Desk Top Unit .....$ 1685.00
QX-10 w/256k MS/DOS Board(run both CP/M and IBM programs) .....$ 2247.00
QX-10 Basic Accounting System w/RX-100 printer .....$ 2450.00
QX-10 Office Productivity System w/FX-100 printer .....$ 3725.00

Rocky Mountain 99'ers

TIC TALK

This publication is printed monthly for the benifit of the membership of the Rocky Mountain 99'ers Computer Club. The Club and the paper are not for the benifit nor backed by any commercial enterprize. Both are non-profit in nature and are for the sole purpose of computer education. Any fees collected are used to defray any cost to maintain the organization. Neither the paper nor the Club have any affiliation with Texas Instruments. Any statements published in this paper are not necessarily the opinion of the membership.

OFFICERS and CHAIRMEN

PRESIDENT	TED MICHELSEN	986-3513
VICE PRESIDENT	MIKE HOLMES	751-7945
SECRETARY	MARTHA WEEG	320-5589
TREASURER	KEN MONSON	233-1788
EDITOR	DAVID OWEN	458-7315
LIBRARIAN	PETE CROWELL	750-5949
MEMBERSHIP	MARTHA WEEG	320-5589
PROGRAM CHAIRMAN	TED MICHELSEN	986-3513
EDITOR/ASSEMBLER SIG	MIKE HOLMES	751-7945
TI FORTH SIG	PETE CROWELL	750-5949
MULTIPLAN SIG	BEN KRAMER	237-1056
THE STAR BOARD BBS		455-3113

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

RETURN TO SENDER:
INT'L LETTER CLASS
MAIL MUST BE ENCLOSED
IN ENVELOPES.

FIRST CLASS

[illegible] Group
P.O. Box 11983
Edmonton, Alberta Canada T5J3L1

****
* 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!!! *
****