```
*--------------------------------------------------------------------------------
* GKLOAD - Gram Kracker Loader Program
*
* GPL Source Code   GRAM 1   >2000
*--------------------------------------------------------------------------------

* Scratch ram equates

CARTSTK  EQU   PAD          Cartridge stack
XMEMFLG  EQU   >830A        Expansion memory flag
CRSREG   EQU   >830B        Cursor/Char register
SELECT   EQU   >830C        Screen address of menu selection char
STLN     EQU   >830E        Char pointer for line editor
INSFLG   EQU   >8310        Insert flag   >00 off  >FF on
AUTOREG  EQU   >8311        Auto repeat delay register
ONEBNK   EQU   INSFLG       Single ROM bank flag
CSFLG    EQU   AUTOREG      Cassette flag
GRMBSE   EQU   >8312        Grom base register
UTILFLG  EQU   >8314        Utility program flag
UTILOP   EQU   >8315        Utility option save flag
GR0      EQU   >8316        Console GRAM flags
GR1      EQU   >8318           .
GR2      EQU   >831A           .

XMLFE    EQU   >831C        XML vector >FE
XMLFF    EQU   >831E        XML vector >FF
TEMP1    EQU   >8340        TEMPORARY register
TEMP2    EQU   >8342        TEMPORARY register
TEMP3    EQU   >8344        TEMPORARY register
TEMP4    EQU   >8346        TEMPORARY register
FAC2     EQU   >834C        Area used by FILES in disk DSR
NMPNTR   EQU   >8356        Name pointer for DSRLNK
KEYBRD   EQU   >8374        Keyboard number to be scanned
KEY      EQU   >8375        Key return from console key scan
TIMER    EQU   >8379        Timer byte decremented every 1/60th of a second

* VDP equates

BOL      EQU   >263          Beginning Of Line in line editor
EOL      EQU   >27B          End Of Line in line editor
VSTACK   EQU   >E00          VDP stack space to save filename
PAB1     EQU   >F00          Disk catalog PAB area
REC      EQU   >F07          Record number of catalog PAB
FILELN1  EQU   >F09          File name length
FILENM1  EQU   >F0A          File name area
PAB2     EQU   >F40          User file PAB area
FILELN2  EQU   >F49          File name length
FILENM2  EQU   >F4A          File name area
CLRTBL   EQU   >0380         Color Table Location
SPRATT   EQU   >0300         Sprite Attribute Table Location
CHRTBL   EQU   >0800         Character table address in VDP
FILEINFO EQU   >0FFA         Header information area
BUFFER   EQU   >1000         Start of 8K buffer

* Misc. equates

SPACE    EQU   >20           Space char
CURSOR   EQU   >1F           Cursor char
FCTN9    EQU   >0F           BACK
ENTER    EQU   >0D           ENTER
```

```
FCTNX    EQU   >0A              FCTN X
RTAROW   EQU   >09              Right arrow
LFAROW   EQU   >08              Left arrow
ERASE    EQU   >07              ERASE
INSERT   EQU   >04              INSERT
DELETE   EQU   >03              DELETE

PERIOD   EQU   >2E

FILERD   EQU   >0500
FILEWT   EQU   >0600
CATOP    EQU   >000D
CATRD    EQU   >020D
CATCL    EQU   >010D
FILES    EQU   >0116            FILES routine in disk DSR
LOWMEM   EQU   >2000            Start of Low Mem-Exp
HIMEM    EQU   >A000            Start of Hi Mem-Exp
EDITOR   EQU   >2BBB            Grom Address for Editor
EDTADD   EQU   >2A30            Cpu Ram Address to move Editor to
EDSTART  EQU   >2A30            Start Address of Editor
UTILST   EQU   >3846->2A30+>2BBB GROM location of UTIL routines
UTILEND  EQU   >3B3C->2A30+>2BBB GROM location of end of UTIL routines
UTILLEN  EQU   UTILEND-UTILST
LSLOAD   EQU   >3000

*--------------------------------------------
* Grom 1 Header >2000
*--------------------------------------------

         GROM 1
         AORG 0

         DATA  >AA55            * Vaildation and Version
         DATA  >0100            * No. Application Programs and Res
         DATA  0                * Power Up Header
         DATA  APPHDR           * Application Program Header
         DATA  0                * DSR Header
         DATA  0                * Subprogram Header
         DATA  0                * Interrupt Header - never in Grom
         DATA  0                * Reserved

*--------------------------------------------
* Application Program Header
*--------------------------------------------

APPHDR   DATA  0
         DATA  START
         STRI  'GRAM KRACKER'

*--------------------------------------------
* Start Gram Kracker Program
* Install pabs
*--------------------------------------------

START
         ST    SPACE,V@FILENM2           Store a space in first char of file n
         MOVE  >19,V@FILENM2,V@FILENM2+1 Fill rest of file name with spaces
         MOVE  10,G@PABS1,V@PAB1         Move PABS into place in VDP RAM
         MOVE  10,G@PABS2,V@PAB2         .
```

```
START1   CALL  INITA     re-entry for editor

*--------------------------------------
* Get Menu Selection from User
*--------------------------------------

GETKEY   MOVE  15,V@>1C7,V@>1C8 Clear selection 5 on return from L/S console
                               * in case its a 99/4 (no - edit memory)
GETKY    CALL  INITB           Re-initialize screen
         ST    @XMEMFLG,@TEMP1
         CALL  MENKEY          Check keyboard
         BR    GETK1           NO FCTN9! Check other keys
         EXIT                  Quit
GETK1    CLR   @CSFLG          Assume no cassette
         DCLR  @UTILFLG        Clear UTILITY program flag
         CASE  @KEY            Branch to selection
         BR    LOADMD          Load Module
         BR    SAVEMD          Save Module
         BR    REINIT          Re-initialize memory
         BR    LSCON           Load console Save/Load routines
         BR    EDIT            Edit Memory

*********************************************************************************

LOADMD   CALL  INPUT           Get file name from user

LOADMD1  CALL  SVFNM           Save file name info
         CLR   @ONEBNK         Clear one bank of ROM flag
         DST   >0001,@TEMP4    Use TEMP4 as a first time through flag

LOADMD2  CALL  RFILE           Read the file
         BS    LOADMD8         ERROR!
         CALL  CLRCASS         Clear screen for cassette

         CEQ   V@FILEINFO,V@FILEINFO+1 Is the first file a UTILITY file?
         BS    UTILLD1         YES! Go load and run UTILITY file

LOADMD2A CHE   9,V@FILEINFO+1  Is it RAM?
         BR    LOADMD4         NO!

LOADMD2B CALL  SELBNK          Put bank select message on screen
         CZ    @ONEBNK         Only one bank of ROM
         BR    LOADMD3         NO!
         ST    '2',V@>271      YES! Change the 1 to a 2
         ST    >FF,@TEMP4      Set 1st time through flag

LOADMD3  CALL  CONTMES         Put press key to continue message up
         CALL  CKON            Make sure write protection is off
         BS    LOADMD9         Write protected!
         CALL  CLRCASS         Clear screen
         MOVE  V@FILEINFO+2,V@BUFFER,@>6000  Move file out to CRAM

         CZ    @ONEBNK         Check one bank flag
         BR    LOADMD6         Taken care of
         ST    >FF,@ONEBNK     Set one bank flag
         CH    9,V@FILEINFO+1  Bank 2 we're loading?
         BS    LOADMD6         YES! Don't loop
         BR    LOADMD2B        Branch around GRAM routines

LOADMD4  CZ    @TEMP4          First time through?
```

```
          BR    LOADMD5          NO!
          CALL  SELBNK
          CALL  CONTMES          YES! Wait for user to press a key

LOADMD5   CALL  CKON                 Check to see if write protection in off
          BS    LOADMD9              NO! Abort
          ST    >FF,@TEMP4           Set 1st time through flag
          DST   V@FILEINFO+4,@TEMP2  Grab GRAM address
          MOVE  V@FILEINFO+2,V@BUFFER,G@0(@TEMP2)  Move file out to GRAM

LOADMD6   CALL  INCFNM           Increment file name if necessary
          BS    LOADMD8          YES! exit

LOADMD7   CEQ   >FF,V@FILEINFO   Last file loaded?
          BS    LOADMD2          NO! Go for more
          CEQ   >80,V@FILEINFO
          BS    UTILLD

LOADMD8   CALL  RSPROT           Put restore write protect message up
LOADMD9   CALL  CLRCASS          Go check for cassette
          CALL  RSFNM            Restore file name to PAB
          CZ    @UTILFLG         Do we need to execute a UTILITY program?
          BS    LOADM10          NO!
          XML   >FE              YES! Go for it
          B     START            Complete restart on return

LOADM10   BR    GETKY            Return to menu

* Load and run UTILITY programs

UTILLD    CALL  RFILE            Load UTILITY file
          BS    UTILLD4          ERROR!
          CALL  CLRCASS          Clear screen

UTILLD1
          DST   V@FILEINFO+4,@TEMP2 Grab load address
          MOVE  V@FILEINFO+2,V@BUFFER,@0(@TEMP2) Move file to memory

          CZ    @UTILFLG         Been through yet?
          BR    UTILLD2          YES!
          INV   @UTILFLG         Set flag
          DST   @TEMP2,@XMLFE    Store start address of routine

UTILLD2   DST   V@FILELN2-1,@TEMP1 Develop file name last char address
          DADD  FILELN2,@TEMP1       .
          CEQ   PERIOD,V*TEMP1   Is it a period?
          BS    UTILLD3          YES! Don't increment file name
          INC   V*TEMP1          Increment file name
UTILLD3   CALL  CKFCTN9          Check for out
          BS    UTILLD4          YEP!
          DCZ   V@FILEINFO       Last file?
          BS    LOADMD8          YES!
          BR    UTILLD           NO! Go look for another file

UTILLD4   CLR   @UTILFLG         ERROR return. Reset flag
          BR    LOADMD8          Return to menu

*****************************************************************************
*
* SAVE MODULE ROUTINE
```

```
*
*********************************************************************************

* Determine what needs to be saved to disk
*  and build an indicator stack in SRAM

SAVEMD   CALL  RSPROT
         BS    GETKY             FCTN9 pressed

BLDSTK   ST    >FF,@DATSTK       Initialize data stack
         DST   >6000,@GRMBSE     Initialize GROM base
         DST   >0400,@TEMP1      Initialize GROM number

BLDSTK1  ST    >20,@TEMP2        Initialize 8K counter
         DST   @GRMBSE,@TEMP3    Store GROM base

BLDSTK2  MOVE  2,G@0(@TEMP3),@TEMP4 Fetch next byte of GROM
         CEQ   @TEMP4,@TEMP4+1   Compare bytes in GROM
         BR    BLDSTK3           Not the same so hop out
         INC   @TEMP3            Increment GROM address
         DEC   @TEMP2            Decrement counter
         BR    BLDSTK2           If not zero then loop
         BR    BLDSTK4           No GROM here so don't push stack

BLDSTK3  PUSH  @TEMP1            Push GROM number onto stack

BLDSTK4  INC   @TEMP1            Increment GROM number by one
         ADD   >20,@GRMBSE       Add next GROM offset to base
         BR    BLDSTK1           If not zero then look for more GROMs

         CLR   @>6000            Page in bank one of ROM

         ST    >60,@GRMBSE       Initialize start address
         ST    >20,@TEMP2        Initialize 8K counter

BLDSTK5  CEQ   *0(@GRMBSE),*1(@GRMBSE)  Compare bytes in ROM
         BR    BLDSTK6           Not the same so hop out
         INC   @GRMBSE           Increment to next address in ROM
         DEC   @TEMP2            Decrement counter
         BR    BLDSTK5           If not done then loop
         BR    BLDSTK9           No ROM so skip page two look up

BLDSTK6  MOVE  >2000,@>6000,V@BUFFER Save 8K for page two look up
         PUSH  @TEMP1            Push stack
         INC   @TEMP1            Increment ROM number

         CLR   @>6002            Page to bank two

         ST    >60,@GRMBSE       Initialize ROM pointer
         ST    >20,@TEMP2        Initialize 8K counter
         DST   BUFFER,@TEMP3     Initialize BUFFER address

BLDSTK7  DCEQ  @0(@GRMBSE),V*TEMP3 Is this ROM the same as bank one
         BR    BLDSTK8           NO! Push page two onto stack
         INC   @GRMBSE           Increment ROM pointer
         INC   @TEMP3            Increment VDP BUFFER pointer
         DEC   @TEMP2            Decrement counter
         BR    BLDSTK7           If not finished then loop
         BR    BLDSTK9           ROM the same as page one so skip
```

```
BLDSTK8  PUSH  @TEMP1           Push page two onto stack

BLDSTK9  CEQ   >FF,@DATSTK      Nothing out there? (Who knows why!?)
         BR    SAVEMD1A         NO! Got something so save it
         CALL  ERRMES           Report error
         BYTE  >8          *    No data error code
         BS    GETKY            Return

* How much space will we need on disk
* This routine must follow BLDSTK because
*  it uses DATSTK to determine space needed

SAVEMD1A ST    @DATSTK,@TEMP4   Store data stack pointer
         INC   @TEMP4           Adjust multipicand
         MUL   34,@TEMP4        Multiply it by 34 sectors/file

         CALL  INPUT            Get the file name from user

         DST   FILENM2,@TEMP1   Initialize PAB2 filename pointer
         CLR   V@FILELN1        Clear PAB1 filename length byte
SAVEMD2  CEQ   SPACE,V*TEMP1    Have we parsed a space in filename
         BS    SAVEMD3          YES! Get out
         INC   V@FILELN1        Increment PAB1 filename length
         CEQ   PERIOD,V*TEMP1   Have we reached a period?
         BS    SAVEMD3          YES! Get out
         DINC  @TEMP1           Bump pointer to next char in PAB2
         BR    SAVEMD2          Parse some more

SAVEMD3  MOVE  V@FILELN1-1,V@FILENM2,V@FILENM1 Transfer device name

* Check for period at end of file name

         DST   V@FILELN2-1,@TEMP1
         DADD  FILELN2,@TEMP1
         CEQ   PERIOD,V*TEMP1
         BS    SAVEMD5

* See if there is enough space on medium

SAVEMD3A DST   CATOP,V@PAB1     Set PAB1 for open catalog
         CALL  CATDSR           Open file
         BS    SAVEMD5
         ST    V@PAB1+1,@TEMP1  Store any errors

         DST   CATRD,V@PAB1     Set PAB1 for read
         CLR   V@REC            Set record number to zero
         CALL  CATDSR           Read a record from catalog
         BS    SAVEMD5
         ST    V@PAB1+1,@TEMP1+1 Store any errors

         DST   CATCL,V@PAB1     Set PAB1 for close
         CALL  CATDSR           Close file
         DCEQ  >0D0D,@TEMP1     Any errors?
         BR    SAVEMD5          YES!

         DST   BUFFER+20,@TEMP1 Set up to point to AU's left field
         ADD   V@BUFFER,@TEMP1+1   .
         MOVE  8,V*TEMP1,@FAC   Move radix 100 number into FAC
         CLR   @FAC+10          Clear error byte
         XML   CFI              Convert radix 100 number to integer
```

```
          CEQ   >03,@FAC+10       Overflow?
          BS    SAVEMD5           YES! Don't do a space check

          DCHE  @TEMP4,@FAC       Is there enough space on disk?
          BS    SAVEMD5           YES!

          FMT                     Put up the insufficient space message
           ROW 17                   .
           COL  3                   .
           HTEX ' ',6,6,' Insufficient Space ',6,6,6
           ROW+ 2
           COL  3
           HTEX ' Press Fctn 9 to cancel or'
          FEND                     .
          CALL  CONTMES
          BS    GETKY

SAVEMD5   CALL  SVFNM             Save file name info
          DST   >FF01,@TEMP4      Load more files flag and filename increment

SAVEMD6   CZ    @DATSTK           Are we at the bottom of the stack
          BR    SAVEMD7           NO!
          CLR   @TEMP4            YES! Assume no UTILITY option save
          CZ    @UTILOP           UTILITY save?
          BS    SAVEMD7           NO!
          ST    >80,@TEMP4        YES! Initialize UTILITY option load flag

SAVEMD7   POP   @TEMP2            POP the stack
          ST    @TEMP2,V@FILEINFO+1 Store memory block number in buffer header
          DEC   @TEMP2            Adjust GROM number

          CHE   >8,@TEMP2         Is it ROM that we're saving?
          BS    SAVEMD8           YES!

          MUL   >20,@TEMP2        Develop GROM base address
          EX    @TEMP2,@TEMP2+1     .
          MOVE  >2000,G@0(@TEMP2),V@BUFFER Move GROM out to buffer
          BR    SAVEM10           Skip around ROM segment

SAVEMD8   CLR   @>6000            Page in bank one of ROM
          CEQ   8,@TEMP2          Is it in fact page one
          BS    SAVEMD9           YES!
          CLR   @>6002            NO! Page in bank two
SAVEMD9   MOVE  >2000,@>6000,V@BUFFER
          DST   >6000,@TEMP2

SAVEM10   ST    @TEMP4,V@FILEINFO Fill in rest of the header info
          DST   >2000,V@FILEINFO+2   .
          DST   @TEMP2,V@FILEINFO+4  .

          CALL  WFILE             Write file out to device
          BS    SAVEM12           Abort on any errors
          CALL  CLRCASS

          CALL  INCFNM            Increment file name if necessary
          BS    SAVEM12           YES! exit

SAVEM11   CEQ   >FF,@TEMP4        Last time through?
          BS    SAVEMD6           NO! Go for another file
```

```
SAVEM12  CALL  CLRCASS
         CALL  RSFNM              Restore File name to PAB2
         BR    GETKY              Go look for another menu selection

* Catalog DSR subprogram

CATDSR   DST   FILELN1,@NMPNTR   Set filename length pointer
         CALL  >10                DSR Link
         BYTE  8
         RTNC

*****************************************************************************
*
* RE-INTIALIZE THE GRAM KRACKER
*
*****************************************************************************

REINIT   CLR   @TEMP4             Initialize loop counter

REINIT1  CALL  SELBNK             Put Select Bank message on screen
         ADD   @TEMP4,V@>271      If 2nd time through make it bank 2

REINIT2  CALL  CONTMES            Put press key to continue message on screen
         BS    GETKY              FCTN9 ?
         CALL  CKON               Make sure write protect is off
         BS    GETKY              If not then ERROR
         CLR   @>6000             Clear rom header byte
         MOVE  >1FFF,@>6000,@>6001  and clear out rom

         INC   @TEMP4             Add to loop counter
         CEQ   >2,@TEMP4          End of the second time ?
         BR    REINIT1            NO! End of first time

         CALL  CLRGRAM            Clear out module grams
                                  * must be a call for trick return
         CALL  RSPROT             Put write protect message up on screen
         B     GETKY              Go look for next selection from user


*****************************************************************************
*
* MEMORY EDITOR LOAD AND EXECUTE
*
*****************************************************************************

EDIT     DST   EDSTART,@XMLFE          Set up for XML >FF
         MOVE  >1444,G@EDITOR,@EDTADD  Move EDITOR to Low Mem-Exp
         XML   >FE                     Link to EDITOR
         BR    START1                  Restart GK program

*****************************************************************************
*
* LOAD/SAVE CONSOLE
*
*****************************************************************************

LSCON    CALL  ACCTON
LSCONA   CALL  INITD
         DST   >0733,@GR0
         DST   >0734,@GR1
```

```
          DST   >0735,@GR2

LGETKY    ST    >05,@TEMP1
          CALL  MENKEY              Check keyboard
          BS    GETKEY              FCTN9 Quit & Clear selection 5
LGETKY1
          CH    >01,@KEY
          BS    GRM012

LSCON1    CLR   @TEMP4
          DST   >3000,@TEMP3
          DST   GR0-2,@TEMP2
          ST    >05,@TEMP4+1
          ADD   @KEY,@TEMP4+1
          DST   >0001,V@FILEINFO
          DST   >2000,V@FILEINFO+2
          DST   >0000,V@FILEINFO+4

LSCON2    DINCT @TEMP2
          DCEQ  GR2+2,@TEMP2
          BS    LSCON3
          CEQ   >07,*TEMP2+1
          BS    LSCON2

          ST    >FF,@TEMP4
          MOVE  9,G@PABS2,V*TEMP3
          ST    @TEMP4+1,V*TEMP3
          FMT
           ROW  17
           COL  19
           HTEX 'Grom/Gram'
          FEND
          ST    *TEMP2+1,V@>23D
          SUB   >03,V@>23D
          CALL  DEVMESS
          CALL  LNEDIT
          BS    LSCONA
          CALL  PARSE
          DINC  @TEMP1
          MOVE  @TEMP1,V@FILELN2,V@9(@TEMP3)
          DADD  40,@TEMP3
          BR    LSCON2

LSCON3    CZ    @TEMP4
          BR    LSCON4
          CALL  ERRMES
          BYTE  >0A             * No Grom Grams selected message
          BS    LSCONA

LSCON4    CALL  RSFNM
          CALL  INITC
          FMT
           ROW  19
           COL  5
           HTEX 'Enable Groms/Grams then'
          FEND
*
* Doug, we need to change the logic here. The Assembly routine must be moved ou
* before the key press is looked for so they can turn off the GK Loader to
* save or load Grom 1.
```

```
*
        MOVE   LSLEN,G@LS,@LSLOAD
        MOVE   UTILLEN,G@UTILST,@>3846
        DST    LSLOAD,@XMLFF
        CALL   MESCONT
        XML    >FF
RETN    BR     LSCON
        CALL   LSERR
        BS     LSCON

GRMO12  MUL    2,@KEY              Devlope Sram address >8316 - >831A
        DADD   >8312,@KEY          .
        CALL   ACCTON
        ST     *KEY+1,V@>C8(@SELECT)
        ST     @KEY+1,@KEY
        INC    @KEY
        EX     *KEY+1,*KEY
        BR     LGETKY
```

```
*****************************************************************************
*
* SUBROUTINE BLOCK
*
*       INIT-Initialization segments with four entry points
*     LNEDIT-Line editor
*      PARSE-Parses input on screen to first space or less char
*      INPUT-Get device.filename from user
*    CONTMES-Put the 'Press and key to continue' message up
*     INCFNM-Increment PAB2 filename if necessary
*     SELBNK-Put select bank message on screen
*     RSPROT-Put restore protection message on screen
*      SVFNM-Push file name info onto stack
*      RSFNM-Pop file name info off of stack
*     CKCASS-Check for cassette file name routine
*
*****************************************************************************

* Set up environment

INITA     ALL   SPACE              Clear the Screen
          MOVE  >08,G@VREGS,#0     Set Up VDP REGs
          ST    >D0,V@SPRATT       Turn Off Sprites
          DST   >64F4,V@CLRTBL     Red Box & white on Dark Blue
          MOVE  >1E,V@CLRTBL+1,V@CLRTBL+2

* Check for Memory Expansion

CHKMEM    ST    3,@XMEMFLG         Allow options 1,2,3 on screen
          DST   @LOWMEM,@TEMP1     Save word at >2000
          DST   >00FF,@LOWMEM      Store mem-check value at >2000
          DCEQ  >00FF,@LOWMEM      Do we have memory expansion?
          BR    CHKME1             NO!
          INC   @XMEMFLG           YES! Allow option 4,5 on screen also
CHKME1    DST   @TEMP1,@LOWMEM     Restore >2000

* Check for 99/4 or 99/4A and load char set

          ST    >05,@KEYBRD        Keyboard 5
          SCAN                     Scan it
          CZ    @KEYBRD            Is It zero (a 99/4 or 4A?)
          BR    L994               Yes, its a 99/4

          CEQ   >04,@XMEMFLG       Do we have memory expansion?
          BR    LDCHAR             NO! Don't allow options 4,5 on the menu
          INC   @XMEMFLG           YES! Allow final option 5 on menu

LDCHAR    DST   >0B00,@FAC         No, its a 99/4A
          CALL  LOCASE             Load Lower Case
          BR    LOADUP             Now go Load Upper case

L994      DST   >0A00,@FAC
          CALL  UPCASE             Load Upper Case into lower case area

LOADUP    CLR   @KEYBRD            4A and 4
          DST   >0900,@FAC
          CALL  UPCASE             Load Upper Case

* Load special chars
```

```
          CLR   V@CHRTBL                          >0800 - >08FF
          MOVE  >F7,V@CHRTBL,V@CHRTBL+1
          MOVE  >18,G@BOX,V@CHRTBL+>30            >00,>01 Chars for Border
          MOVE  >08,G@CRSDEF,V@CHRTBL+>F8         >1F Cursor char

          RTN

* Put up Red Box and Menu Screen

INITB     HOME                YPT=0 XPT=0
          FMT
            HCHAR 32,7        Top
            VCHA  22,7        Left
            HCHA  31,7        Bottom
            VCHA  25,7        Right
            ROW   4
            COL   1
            HCHA  30,6        Thin Line
          FEND

          CALL INITC
          FMT
            ROW   >02
            COL   >05
            HTEX  'G R A M   K R A C K E R'
            ROW   >15
            COL   >04
            HTEX  >08,'1985 by Millers Graphics'
            ROW   >06
            COL   >08
            HTEX  '1  Load Module '
            ROW+  >02
            COL   >08
            HTEX  '2  Save Module '
            ROW+  >02
            COL   >08
            HTEX  '3  Init Module Space'
          FEND
          CH    >3,@XMEMFLG         Expansion memory?
          BR    ENDSUB1             NO! So don't put rest of menu up
          FMT
            ROW+  >02
            COL   >08
            HTEX  '4  Load/Save Console'
          FEND
          CH    >4,@XMEMFLG         99/4?
          BR    ENDSUB1             YES! Don't put EDITOR selection up
          FMT
            ROW+  >02
            COL   >08
            HTEX  '5  Edit Memory'
          FEND

ENDSUB1   RTN

* Clear screen on cassette routine detection

CLRCASS   CZ    @CSFLG
          BS    INITC
          ALL   SPACE
```

```
INITC    FMT
           ROW   >10
           FOR   >05
             ROW+  >01
             COL   >03
             HCHAR 28,SPACE
           FEND
         FEND
         RTN

INITD    FMT
          ROW  6
          COL  8
          HTEX '1  Load Console'
          ROW+ 2
          COL  8
          HTEX '2  Save Console'
          ROW+ 2
          COL  8
          HTEX '3  Grom/Gram 0      '
          ROW+ 2
          COL  8
          HTEX '4  Grom/Gram 1      '
          ROW+ 2
          COL  8
          HTEX '5  Grom/Gram 2'
         FEND
         BR   INITC

*****************************************************************************
*
*      LINE EDITOR (GPL)   193 bytes in length
*
*      Features: Flashing cursor
*                Auto-repeat
*                Right and left arrow key movement
*                Insert and delete capability
*                Erase line feature
*                BACK and ENTER return to caller
*
*****************************************************************************

* Start of line editor

LNEDIT   CALL  ACCTON           Sound a beep at user upon entry
LNEDITA  ST    CURSOR,@CRSREG   Store cursor char in cursor/char register
         DST   BOL,@STLN        Initialize line register with BOL
         DCLR  @INSFLG          Reset insert mode & auto-repeat timer

* Flash cursor

LNEDIT1  CLR   @TIMER           Clear GPL timer
         EX    @CRSREG,V*STLN   Exchange cursor with whatever is on the screen

* Scan keyboard, check timer and auto-repeat

LNEDIT2  SCAN                   Do a key scan
         BS    REPEAT1          Got a key so service
          INC   @AUTOREG        Increment auto-repeat timer
          CEQ   >FF,@KEY        Is a key still down?
```

```
            BS    REPEAT          NO! Go check cursor flash
             CHE   >80,@AUTOREG   Is it time to start auto-repeating?
             BR    REPEAT         NO! Go check cursor flash
             SUB   >1E,@AUTOREG   YES! Set cursor movement speed
             BR    ENDRPT         Service user input
REPEAT      CHE   >0F,@TIMER     Are we out of time
            BR    LNEDIT2        NO! Go do another scan
            BR    LNEDIT1        YES! Flash cursor

* Got a key. Take cursor off screen

REPEAT1  CLR   @AUTOREG         Clear the auto-repeat register
ENDRPT   CEQ   CURSOR,@CRSREG   Got a key. Is cursor on the screen?
         BS    LNEDIT4          NO! No need to restore screen char
         EX    @CRSREG,V*STLN   YES! Exchange the cursor for char

* What do we have?

LNEDIT4  CHE   SPACE,@KEY       Was key pressed less than a space?
         BR    LNEDIT5          YES! Must be a FCTN key
         CZ    @INSFLG          Are we in INSERT mode?
         BR    LNEDI10          YES! Insert key press into string

         ST    @KEY,V*STLN      Put char onto the screen
         BR    LNADVN           Branch to sub to advance to next space

* BACK routine

LNEDIT5  CLR   @INSFLG
         CEQ   FCTN9,@KEY       Was it FCTN BACK?
         BS    ERRTN            YES! Return

* RIGHT ARROW routine

         CEQ   RTAROW,@KEY      Was it the right arrow?
*        BR    LNEDIT6          See if it is the left arrow
         BS    LNADVN           Branch to sub to advance to next space

* LEFT ARROW routine

LNEDIT6  CEQ   LFAROW,@KEY      Is it the left arrow key?
         BR    LNEDIT7          NO!
         DCEQ  BOL,@STLN        Are we at the beginning of the line
         BS    LNERROR          YES! Tell the user
         DDEC  @STLN            NO! Backup a space
         BR    LNEDIT1          Return to line editor

* ENTER routine

LNEDIT7  CEQ   ENTER,@KEY       Is it the ENTER key?
         BR    LNEDIT8          NO!
LNRTN    RTN                    YES! Return to caller

* DELETE routine

LNEDIT8  CEQ   DELETE,@KEY      Is it the DELETE key?
         BR    LNEDIT9          NO!
         DST   EOL+2,@TEMP1     Store one position past the EOL
         DSUB  @STLN,@TEMP1     Calculate length to move
         MOVE  @TEMP1,V@1(@STLN),V*STLN Move end of string in one position
```

```
          BR    LNEDIT1

* INSERT routine

LNEDIT9   CEQ   INSERT,@KEY      Is it the INSERT key?
          BR    LNEDI12          NO!
          ST    >FF,@INSFLG      Set INSERT flag
          BR    LNEDIT1          Continue

LNEDI10   DST   EOL,@TEMP1       Fetch address of EOL
LNEDI11   ST    V*TEMP1,V@1(@TEMP1) Move a char over one position
          DDEC  @TEMP1           Decrement pointer
          DCH   @TEMP1,@STLN     Have we reached the cursor position?
          BR    LNEDI11          NO! Go move another char over
          ST    @KEY,V*STLN      Put key input into vacant space
          BR    LNADVN           Advanve cursor

* ERASE line routine

LNEDI12   CEQ   ERASE,@KEY       Is it the ERASE key?
          BR    LNEDI13          NO! Sound an error
          MOVE  27,V@BOL-1,V@BOL Clear input line
          BR    LNEDITA          Restart line editor

* FCTN X UTILITY save routine

LNEDI13   CEQ   FCTNX,@KEY
          BR    LNERROR
          CEQ   '2',V@264         Are we calling from the LOAD option?
          BS    LNERROR           NO! Should be a square here so skip
          CALL  ACCTON
          INV   @UTILOP

          MOVE  11,V@562,V@563

          CZ    @UTILOP
          BS    LNEDIT1
          FMT
           ROW 17
           COL 19
           HTEX '[Util Opt]'
          FEND
          BR    LNEDIT1


* LINE EDITOR subroutines

LNERROR   CALL  BADTON           Sound a bad tone for user
          BR    LNEDIT1          Return to line editor

LNADVN    DCH   EOL,@STLN        Are we at the end of the line?
          BS    LNERROR          YES! Don't go any further

          DINC  @STLN            NO! Increment to next space
          BR    LNEDIT1          Do it all again
* End of line editor

* Parse to end of line routine

PARSE     DST   BOL,@TEMP1       Fetch BOL address
```

```
PARSE1  CH    SPACE,V*TEMP1   Are we on a space char of less?
        BR    PARSE2          YES! Done
        DINC  @TEMP1          Move ahead one
        BR    PARSE1          Check where we are again
PARSE2  DSUB  BOL,@TEMP1
        ST    @TEMP1+1,V@FILELN2 Install new length of line into PAB2
        MOVE  @TEMP1,V@BOL,V@FILENM2 Move new file name into PAB2
        RTN                   Return

* Input Device.Filename routine

DEVMESS FMT                     Put promp on screen
        ROW  17
        COL  3
        HTEX 'Device.Filename'
       FEND
       RTN

INPUT   CALL  DEVMESS
*       DST   V@FILELN2-1,@TEMP1 Grab file name length from PAB2
*       MOVE  @TEMP1,V@FILENM2,V@BOL Move file name to screen
        MOVE  V@FILELN2-1,V@FILENM2,V@BOL Move file name to screen
        CALL  LNEDIT            Line editor
        BS    GETKY
        CALL  PARSE             Parse to end of line
        CALL  CKCASS            Check for cassette
        CALL  CLRCASS           Clear appropriate screen space
        RTN

* Press any key to continue message

MESCONT CALL ACCTON

MESCON1 FMT
         ROW 21
         COL  3
         HTEX 'Press space bar to continue'
        FEND
        RTN

CONTMES CALL MESCONT

CONT    ST    >FF,@TEMP1        Indicate a CALL CONT for menky1
MENKEY
        MOVE WTLEN,G@WAITRT,@XMLFF  Move wait routine into SRAM
        XML  >FF                    Branch to key scan routine
        CEQ  FCTN9,@KEY         Has Fctn9 been pressed
        BS   ERRTN              YES! return with cond bit set
        SUB  >31,@KEY           NO! Make key 0-4
        CHE  @TEMP1,@KEY        Check max for this menu
        BR   MENKY1             Key is ok
        CALL BADTON             Bad key
        BR   MENKEY             Go get another
MENKY1
        CALL INITC              Clear bottom of screen
        CEQ  >FF,@TEMP1         Was this a CALL CONT ?
        BS   ENDCONT            YES! were done
        ST   @KEY,@SELECT+1     NO! a menu selection
        DSLL >06,@SELECT
        ST   >07,V@>C8(@SELECT) Put box on top of number
```

```
ENDCONT RTN

* File write routine

WFILE    DST  FILEWT,V@PAB2     Reset PAB2 for write
         FMT
           ROW   19
           COL   8
           HTEXT 6,6,6,' Saving ',6,6,6
         FEND
         BR   RFILE1            Write file

* File read routine

RFILE    DST  FILERD,V@PAB2     Reset PAB2 for read
         FMT
           ROW   19
           COL   8
           HTEXT 6,6,6,' Loading ',6,6,6
         FEND

RFILE1   DST  FILELN2,@NMPNTR   Set up name length pointer
         CALL >10               Write to device
         BYTE 8
         BS    DSRMES            NO DEVICE error
         CZ    V@PAB2+1          Any other errors?
         BR    DSRMES            YES!
         RTN

* Error reporting routine. Returns with COND bit set

DSRMES   ST    V@PAB2+1,@TEMP1   Get error from PAB
         SRL   5,@TEMP1          Shift it to a byte boudary
LSERR    CALL  CLRCASS
DSRMES1  FMT                     Put up error message
          ROW  17
          COL  8
          HTEX 6,6,6,' I/O ERROR ',6,6,6
         FEND
         BR    ERRMES2

ERRMES   CALL  CLRCASS
         FETCH @TEMP1              Get error message number for non DSR errors
ERRMES1  FMT
          ROW  17
          COL  10
          HTEX 6,6,6,' ERROR ',6,6,6
         FEND

* Display message in center of row

ERRMES2  CALL BADTON

         MUL   >2,@TEMP1
         MOVE  >02,G@ERRTBL(@TEMP1),@TEMP2       get address of len. from table
         MOVE  >01,G@0(@TEMP2),@TEMP1+1          get length
         DST   @TEMP1,@TEMP3                     copy it for col index
         DINV  @TEMP3                            neg. for odd len and col index
         DSRA  >01,@TEMP3                        adjust it for col index
         MOVE  @TEMP1,G@1(@TEMP2),V@>271(@TEMP3) put up message
```

```
          CALL MESCON1            Put up Press space bar to continue message
          CALL CONT                 .

ERRTN     CEQ  @PAD,@PAD          Set COND bit
ERRMES3   RTNC
*

INCFNM    DST   V@FILELN2-1,@TEMP3 Get filename length
          ST    @TEMP3+1,@TEMP1   Copy length into TEMP1
          DADD  FILELN2,@TEMP3    Develop end of filename address
          DST   @TEMP3,@TEMP2     Copy address
          CEQ   PERIOD,V*TEMP3    Is it a period?
          BS    INCFNM2           YES! Skip increment
          CEQ   1,@TEMP4+1        First time through?
          BR    INCFNM1           NO!

BPARSE    CLR   @TEMP1+1          Clear length counter
BPARS1    CEQ   PERIOD,V*TEMP2    Is char a period?
          BS    BPARS2            YES! Done
          DDEC  @TEMP2            Decrement address pointer
          INC   @TEMP1+1          Increment length counter
          DEC   @TEMP1            Decrement total length counter
          BR    BPARS1            If not at end then loop
BPARS2    CHE   >A,@TEMP1+1       Is filename 10 or greater in length?
          BS    BPARS3            YES!
          INC   V@FILELN2         NO! Increment filename length
          DINC  @TEMP3
BPARS3    EQU   $                 Return

INCFNM1   ST    @TEMP4+1,V*TEMP3  Increment file name
          ADD   >30,V*TEMP3       Make it ASCII
          INC   @TEMP4+1          Prepare next increment
INCFNM2   CALL  CKFCTN9
          RTNC

*

SELBNK    FMT                     Put enable bank message up
           ROW 19
           COL  5
           HTEX 'Enable bank 1 and then'
          FEND
          RTN

*

RSPROT    FMT                     Put up restore write protect message
           ROW 19
           COL  4
           HTEX 'Restore write protect and'
          FEND
          CALL  CONTMES
          RTNC

*

SVFNM     MOVE  27,V@FILELN2,V@VSTACK Save file name
          RTN
```

```
RSFNM    MOVE  27,V@VSTACK,V@FILELN2 Restore file name
         RTN

*

CKON     DST   @>6000,@TEMP1     Save CRAM location
         DST   @TEMP1,@>6000     Check to see if we're in the right bank
         DST   @>6000,@TEMP1        .
         DINV  @>6000            Try to invert it
         DCEQ  @>6000,@TEMP1     Write protected?
         BS    CKON1             YES!
         DST   @TEMP1,@>6000     NO! Restore old value
         RTN

CKON1    CALL  ERRMES            Put error message up on screen
         BYTE  >09       *       Error message 9
         RTNC                    Return with COND set

* Check for cassette routine

CKCASS   CLR   @CSFLG            Clear cassette flag
         CH    >4,V@FILELN2      File name >4 chars in length?
         BS    CKCASS1           YES!
         DCEQ  'CS',V@FILENM2    First two chars CS?
         BR    CKCASS1           NO!
         INV   @CSFLG            Assume cassette and set flag
         ST    4,V@FILELN2       Make sure file length is 4
         ST    PERIOD,V@FILENM2+3 Make sure there is a period after CS1
CKCASS1  RTN

*

CKFCTN9  SCAN
CKFCTN91 CEQ   FCTN9,@KEY
         RTNC

*

CLRGRAM  MOVE  1,@TEMP4+1,G@>FFFF   Set up for trick RTN after XML
         MOVE  CLGRLN,G@CLGR,@XMLFF Move assembly program out
         XML   >FF                  Execute it to clear out Gram space
*              no RTN needed
*              its already at >FFFF

*********************************************************************
*
* Data Tables
*
*********************************************************************

VREGS    BYTE  >00,>E0,>00,>0E,>01,>06,>00,>F5

PABS2    BYTE  >05,>00,>0F,>FA,>00,>00,>20,>06,>00
PABS1    BYTE  >00,>0D,>10,>00,>26,>26,>00,>00

BOX      BYTE  >00,>00,>FF,>FF,>FF,>00,>00,>00
         BYTE  >FF,>FF,>FF,>FF,>FF,>FF,>FF,>FF

CPYRHT   BYTE  >3C,>42,>99,>A1,>A1,>99,>42,>3C
```

```
CRSDEF BYTE  >00,>7E,>42,>42,>42,>42,>42,>7E

ERRTBL   DATA  NODEV,WRPROT,BDOATR,ILLOP
         DATA  DSKFLL,PEOF,DVCER,FILERR
         DATA  NODAT,CNTWT,NOSEL

NODEV    STRI 'Bad device name'

WRPROT * STRI 'Device write protected'
         STRI 'Write protection'

BDOATR   STRI 'Bad open attribute'

ILLOP    STRI 'Illegal operation'

DSKFLL * STRI 'Device out of space'
         STRI 'Out of space'

PEOF     STRI 'End of file'

DVCER  * STRI 'Device or medium error'
         STRI 'Device error'

FILERR   STRI 'File error'

NODAT    STRI 'No data to save'

CNTWT    STRI 'Write protect active'

NOSEL    STRI 'No Grom/Gram selected'

*****************************************************************************
*
* ASSEMBLY ROUTINES
*
*****************************************************************************

WAITRT             *        AORG >831E
                   * STATUS EQU  >837C
                   * SCAN   EQU  >000E
                   * KEY    EQU  >8375
       DATA >8320  *        DATA >8320
       DATA >0300  * LOOP   LIMI 2                  Enable Interrupts
       DATA >0002  *
       DATA >0300  *        LIMI 0
       DATA >0000  *
       DATA >06A0  *        BL   @SCAN              Look for key
       DATA >000E  *
       DATA >D020  *        MOVB @STATUS,R0         New key ?
       DATA >837C  *
       DATA >13F7  *        JEQ  LOOP               NO!
       DATA >9820  *        CB   @LOOP+3,@KEY       YES! Is it Fctn4 ?
       DATA >8323  *
       DATA >8375  *
       DATA >13F3  *        JEQ  LOOP               YES!
       DATA >0460  *        B    @>6A               NO! Back to GPL
       DATA >006A  *
WTLEN  EQU  $-WAITRT
```

```
CLGR                    * GWA     EQU  >0402
                        * GWD     EQU  >0400
                        * ROLB    EQU  >83E1
        DATA >8320      *
        DATA >0200      *         LI   R0,>6000          Set up Gram address
        DATA >6000      *
        DATA >DB40      *         MOVB R0,@GWA(R13)      .
        DATA >0402      *
        DATA >DB60      *         MOVB @ROLB,@GWA(R13)   .
        DATA >83E1      *
        DATA >0402      *
        DATA >0540      *         INV  R0                r0 now = >9FFF
        DATA >04C1      *         CLR  R1
        DATA >DB41      * LOOP    MOVB R1,@GWD(R13)      Clear out Module Gram space
        DATA >0400      *
        DATA >0600      *         DEC  R0                .
        DATA >16FC      *         JNE  LOOP              Not done yet
        DATA >045B      *         RT                     Fini, go back to GPL
CLGRLN  EQU $-CLGR      *

                        *         AORG >3000
                        * GPLWS   EQU  >83E0
                        * GRADD   EQU  >3A3E
                        * TEMP1   EQU  >8340
                        * TEMP4   EQU  >8346
                        * VDPWD   EQU  >8C00
                        * VDPRD   EQU  >8800
                        * FLINFO  EQU  >FFA
                        * XVSBW   EQU  >3956
                        * XVMBW   EQU  >396E
                        * H2000   EQU  >3ADC
                        * STATUS  EQU  >837C
                        * GWADD   EQU  >3A2C
                        * SCANCT  EQU  >3940
                        * NAMBUF  EQU  >3E70
                        * XVMBR   EQU  >397E
                        * PNTR    EQU  >8356
                        * XDSRLN  EQU  >3846
                        * VWDADD  EQU  >398E
                        * VRDADD  EQU  >3992
                        * BUFFER  EQU  >1000
                        * RETURN  EQU  >1234        Dummy value
LS      DATA >02E0      * LS      LWPI WSR                 Load workspace pointer
        DATA >310A      *
        DATA >0300      * LOOP    LIMI 2
        DATA >0002      *
        DATA >0300      *         LIMI 0
        DATA >0000      *
        DATA >06A0      *         BL   @SCANCT             Look for a column 0 press
        DATA >3940      *
        DATA >0546      *         INV  R6                  Make it data true
        DATA >0246      *         ANDI R6,>200             Make sure we're only looking
        DATA >0200      *
        DATA >13F6      *         JEQ  LOOP                NO key so loop
        DATA >06A0      * LOOPB   BL   @SCANCT             Look for a column 0 press
        DATA >3940      *
        DATA >0546      *         INV  R6                  Make it data true
        DATA >16FC      *         JNE  LOOPB               Keep looping until no key
        DATA >C188      *         MOV  R8,R6               Initialize R6
        DATA >9814      * LS1     CB   *R4,@R6LB           Load/Save this GROM?
```

```
DATA >3117 *
DATA >161A *          JNE   LS3                 NO!
DATA >91A0 *          CB    @TEMP4+1,R6         Save operation?
DATA >8347 *
DATA >160B *          JNE   LS2                 NO!
DATA >0200 *          LI    R0,BUFFER+>4000     Make R0 the VDP buffer addres
DATA >5000 *
DATA >06A0 *          BL    @SETADD             Set VDP and GROM addresses
DATA >30D6 *
DATA >D810 * LOOP1    MOVB  *R0,@VDPWD
DATA >8C00 *
DATA >0601 *          DEC   R1                  Decrement loop counter
DATA >16FC *          JNE   LOOP1               Loop until done
DATA >06A0 *          BL    @SETDSR             Write file to device
DATA >30B6 *
DATA >100C *          JMP   LS3
DATA >06A0 * LS2      BL    @SETDSR             Read file in
DATA >30B6 *
DATA >0200 *          LI    R0,BUFFER           Make R0 the VDP buffer addres
DATA >1000 *
DATA >06A0 *          BL    @SETADD             Set VDP and GROM addresses
DATA >30D6 *
DATA >0220 *          AI    R0,>400             Add GROM WD offset
DATA >0400 *
DATA >D420 * LOOP2    MOVB  @VDPRD,*R0
DATA >8800 *
DATA >0601 *          DEC   R1                  Decrement counter
DATA >16FC *          JNE   LOOP2               Loop until done
DATA >0225 * LS3      AI    R5,>0120            Adjust GROM # and address fo
DATA >0120 *
DATA >0607 *          DEC   R7                  Done?
DATA >1311 *          JEQ   DONE                YES!
DATA >05C4 *          INCT  R4
DATA >0200 *          LI    R0,FLINFO+1         Load R0 with GROM # address
DATA >0FFB *
DATA >C045 *          MOV   R5,R1               Load R1 with GROM #
DATA >06A0 *          BL    @XVSBW              Write GROM # out to file hea
DATA >3956 *
DATA >0200 *          LI    R0,FLINFO+4         Load R0 with GROM address in
DATA >0FFE *
DATA >06C1 *          SWPB  R1                  Position MSB of GROM address
DATA >06A0 *          BL    @XVSBW              Write GROM address to header
DATA >3956 *
DATA >10D3 *          JMP   LS1
DATA >D820 * ERROR    MOVB  @H2000,@STATUS      Indicate an error
DATA >3ADC *
DATA >837C *
DATA >D800 *          MOVB  R0,@TEMP1           Move error code into TEMP1 f
DATA >8340 *
DATA >0200 * DONE     LI    R0,RETURN           Restore GROM address
DATA RETN   *
DATA >06A0 *          BL    @GWADD              Set return address
DATA >3A2C *
DATA >0200 *          LI    R0,>264
DATA >0264 *
DATA >0201 *          LI    R1,MESSAG
DATA >30F0 *
DATA >0202 *          LI    R2,26
DATA >001A *
DATA >06A0 *          BL    @XVMBW
```

```
DATA >396E *
DATA >06A0 * LOOP3  BL   @SCANCT          Look for a column 0 press
DATA >3940 *
DATA >0546 *        INV  R6               Make it data true
DATA >13FC *        JEQ  LOOP3            NO key so loop
DATA >06A0 * LOOP4  BL   @SCANCT          Look for a column 0 press
DATA >3940 *
DATA >0546 *        INV  R6               Make it data true
DATA >16FC *        JNE  LOOP4            Keep looping until no key
DATA >02E0 *        LWPI GPLWS            Reload GPL workspace
DATA >83E0 *
DATA >0460 *        B    @>70             Return
DATA >0070 *
DATA >C28B * SETDSR MOV  R11,R10          Save return address
DATA >C003 *        MOV  R3,R0            Load R0 with PAB name length
DATA >0201 *        LI   R1,NAMBUF-1      Load R1 with buffer address
DATA >3E6F *
DATA >0202 *        LI   R2,27            Transfer 27 bytes
DATA >001B *
DATA >06A0 *        BL   @XVMBR                 .
DATA >397E *
DATA >C803 *        MOV  R3,@PNTR         Load PNTR with name length ac
DATA >8356 *
DATA >0420 *        BLWP @XDSRLN          Link with device
DATA >3846 *
DATA >13D6 *        JEQ  ERROR            ERROR!
DATA >0223 *        AI   R3,>28           Increment file name length pc
DATA >0028 *
DATA >045A *        B    *R10             Return
DATA >C28B * SETADD MOV  R11,R10          Save return address
DATA >06A0 *        BL   @VRDADD          Set VDP address
DATA >3992 *
DATA >04C0 *        CLR  R0               Zero out R0
DATA >D020 *        MOVB @R5LB,R0         Make R0 the GROM address
DATA >3115 *
DATA >06A0 *        BL   @GWADD           Set GROM address to be moved
DATA >3A2C *
DATA >C020 *        MOV  @>83FA,R0        Grab GROM library base
DATA >83FA *
DATA >0201 *        LI   R1,>2000         Initialize counter
DATA >2000 *
DATA >045A *        B    *R10             Return
TEXT 'Restore Op Sys and Loader '
           * R5LB   EQU  $+11
           * R6LB   EQU  $+13
DATA >0000 * WSR    DATA 0,0,0
DATA >0000 *
DATA >0000 *
DATA >3009 *        DATA >3009       R3   File name length pointer
DATA >8317 *        DATA >8317       R4   Grom flag address
DATA >0100 *        DATA >0100       R5   GROM # and MSB of GROM addres
DATA >0607 *        DATA >0607       R6   Write flag and GROM select fl
DATA >0003 *        DATA >0003       R7   Counter
DATA >0607 *        DATA >0607       R8

LSLEN  EQU  $-LS  *
```

```
*
*--------------------------------------------------------------------------------
* QUIT ROUTINE
*
ASCHQT  BL    @SCANCT           Scan keyboard column 0
        CZC   @H4000,R6         Is the CTRL key down?
        JNE   ASCHEX            NO! Must be ASC/HEX toggle
*
        MOV   @H2000,R0         Move GROM 1 base address into R0
        BL    @GWADD            Set address for GROM 1
        BL    @GRDAT            Grab the first word from GROM 1
        SWPB  R1                   .
        BL    @GRDAT               .
        C     R1,@H55AA         Is it the GRAMKRACKER flag word?
        JNE   ASCH11            NO! Don't exit
*
        MOV   @SAVGRM,R0        Restore previous GROM address
        BL    @GWADD               .
        LI    R0,>81A0          Load VDP Register 1 with graphics/screen off
        BL    @VRDADD              .
        LWPI  GPLWS             Load GPL workspace
        B     @>6A              Return with COND bit reset
*
ASCHEX  XOR   @H100,R15         Toggle ASCII/HEX flag
ASCH11  BL    @CROFF            Turn cursor off
        BL    @MEMBLK           Rewrite memory block
        COC   @HC,R15           Are we in the MEDIT?
        JEQ   ASCHE1            YES!
        COC   @H8,R15           Are we in SEARCH
        JNE   ASCHE1            NO!
        MOV   @TEMP22,@BLKSCN   Rewrite memory block
        MOV   @TEMP23,@BYTENO      .
        MOV   @TEMP26,@MADDR       .
        MOV   @TEMP27,@CGV         .
        BL    @MEMBLK              .
        MOV   @TEMP20,@BLKSCN   Restore SEARCH values
        MOV   @TEMP24,@BYTENO      .
        LI    R7,SRCH$             .
        MOV   R7,@MADDR            .
        MOV   @HAC,@CGV            .
ASCHE1  LI    R2,-1             Put a -1 in R2 for CURSOR. This is in case the
*                                cursor falls on an EDGE character after togglin
        B     @EDIT3            Re-enter key scan through EDIT3
```

```
*
*--------------------------------------------------------------------------------
* Basic offset switch
*
BSWTCH XOR  @H1000,R15        Toggle basic offset bit
*
BOFSET LI   R0,TOP+121        Move address down a line
       LI   R1,DLINE          Load R1 with double line character
       LI   R2,35             Thirty-five characters to write
       BL   @RPTBLK           Put double line on top of memory area
       COC  @H1000,R15        Is the basic offset flag set?
       JNE  BOFSE1            NO!
       LI   R0,TOP+132        Add 51 to it
       LI   R1,BMESEG         Load message string address
       LI   R2,13             Ten bytes to write
       BL   @VMBW             Put message on the screen
BOFSE1 EQU  $
       COC  @H100,R15         Is ASCII on the screen?
       JNE  BSWTC1            NO! Must be hex so skip
       B    @ASCH11           Rewrite the screen
BSWTC1 B    @POPSTK           Return through POPSTK
```

```
COLOR   INCT @COLPNR          Increment color stack pointer
ERRCOL  MOV  R11,R9           Save return
        MOV  @COLPNR,R1       Grab color pointer
        MOV  *R1,@TEMP1       Move color into TEMP1
DCOUNT  EQU  $                >200
        LI   R0,>87           Load register 7
        MOVB @TEMP1+1,R0      Move error colors into R0
        JNE  ERRCO1           Non zero then we are not at end of stack
        LI   R0,COLORS        Start of stack
        MOV  R0,@COLPNR       Reload stack pointer
        JMP  ERRCOL           Go through above again
ERRCO1  LI   R1,FCTN7         Load FCTN7
        CB   R1,@KEYBRD       Are we really doing a color change?
        JEQ  ERRCO3           YES! Don't load error colors
        BL   @VRDADD+2        Set screen up with new colors

ERRCO2  BL   @CHKKEY          Check key to see if keyboard is free
        CB   R1,@HBFF         No key depressed?
        JNE  ERRCO2           NO! Not yet; loop
ERRCO3  LI   R0,>87           Load register 7
        MOVB @TEMP1,R0        Move screen color into R0
        BL   @VRDADD+2        Set screen colors
        B    *R9              Return to caller
```

```
TOP     EQU  1
*
UTILWS  EQU  $
UTIL0   EQU  UTILWS+1
UTIL1   EQU  UTILWS+3
SCRN    DATA TOP+42,34
        TEXT 'c3E70    Start 0000    Finish 0000'
        DATA TOP+91,24
        TEXT 'Dest c0000    Fill   00 '
        DATA TOP+722,34
        TEXT 'Wndow 1  Pg up 4  Color 7  Bias  0'
        DATA TOP+762,34
        TEXT 'Move  2  Srch  5  Dump  8  AscHx ='
        DATA TOP+802,34
        TEXT 'Fill  3  Pg dn 6  Back  9  Hm ENTR'
        DATA TOP+882,34
        TEXT 'Device name c3E70          ERROR ok'
        DATA 0
*
FLEN1   EQU  0
FLEN2   EQU  >400
FLEN3   EQU  >800
FLEN4   EQU  >C00
*
        DATA 0
TEMP1   DATA 0
TEMP2   DATA 0              RESERVED FOR R14/MADDR DIFF
TEMP3   DATA TOP+161        RESERVED FOR MEMBLK REAL CURSOR ADDR
FIELD   DATA TOP+42+FLEN1       CGV
        DATA CGVSTK,CGVCHK
CGV     TEXT 'c '
        DATA TOP+43+FLEN4       MADDR
        DATA HEXSTK,RANGEA
MADDR   DATA >3E70
        DATA TOP+57+FLEN4       Aop
        DATA HEXSTK,RANGEA
AOP     DATA >0000
        DATA TOP+72+FLEN4       Bop
        DATA HEXSTK,RANGEA
BOP     DATA >0000
*
        DATA TOP+96+FLEN1       DCGV
        DATA CGVSTK,CGVCHK
DCGV    TEXT 'c '
        DATA TOP+97+FLEN4       Dop
        DATA HEXSTK,RANGEB
DOP     DATA >0000
LSTFLD  DATA TOP+112+FLEN2      Fop
        DATA HEXSTK,0
FOP     DATA >0000
        DATA 0
*
HEXSTK  BYTE >80
        TEXT '0123456789'
        TEXT 'ABCDEF'
        TEXT 'abcdef'
        BYTE >FF
CGVSTK  BYTE >20
        TEXT 'CGVcgv'
HBFF    BYTE >FF
```

```
        EVEN
*
H10     DATA >10
HG      DATA >6700
HV      DATA >7600
HBA     DATA >A00
H8000   DATA >8000
H55AA   DATA >55AA
SPEED   DATA >1800
SRCH$   DATA 0,0,0,0,0,0
*
SAVGRM  DATA 0
XCHARS  DATA 0,0,>FF00,0 -
        DATA 0,>00FF,>00FF,0 =
        DATA 0,0,0,0 edge
        DATA 0,0,>2000,0 tic
        DATA >0078,>7878,>7878,>7878 cursor
BMESEG  TEXT ' basic  bias '
OK      TEXT 'ok'
PAB     DATA >0012,>1000,>5047,0,>0000
WSR     BSS  32
*--------------------------------
KEYBRD  EQU  >8375
STATUS  EQU  >837C
GPLWS   EQU  >83E0
R3LB    EQU  GPLWS+7
R12LB   EQU  GPLWS+25
VDPWD   EQU  >8C00
VDPRD   EQU  >8800
VDPWA   EQU  >8C02
VDPSTA  EQU  >8802
GRMRA   EQU  >9802
GRMWA   EQU  >9C02
GRMRD   EQU  >9800
GRMWD   EQU  >9C00
SOUND   EQU  >8400
OFFSET  EQU  ->6000
UPAROW  EQU  >B00
RTAROW  EQU  >900
DNAROW  EQU  >A00
LFAROW  EQU  >800
LFBRAK  EQU  >7B00
FCTN    EQU  H1000
FCTN1   EQU  >300
FCTN2   EQU  >400
FCTN3   EQU  >700
FCTN4   EQU  >200
FCTN5   EQU  >E00
FCTN6   EQU  >C00
FCTN7   EQU  >100
FCTN8   EQU  >600
FCTN9   EQU  >F00
FCTN0   EQU  >BC00
FCTNEQ  EQU  >500
ENTER   EQU  >D00
CRSOR   EQU  >1F00
TCOUNT  EQU  >180
SPDSET  EQU  >1800
*
COLPNR  DATA COLORS
```

```
COLORS  DATA >F44F              White/D Blue
        DATA >1221              Black/M Grn
        DATA >FCCF              White/M Grn
        DATA >1771              Black/L Blue
        DATA >F11F              White/Black
        DATA 0
*
WSTACK  TEXT 'g '
        DATA >6000
WSTAC1  TEXT 'v '
        DATA >F40
*
KEY14   DATA RTAROW,RTKEY
        DATA LFAROW,LFKEY
        DATA FCTNEQ,ASCHQT
        DATA FCTN0,BSWTCH
        DATA ENTER,RTKEY1
        DATA FCTN7,COLOR
        DATA 0

KEY1    DATA FCTN1,SWPMEM
        DATA FCTN2,MOVER
        DATA FCTN3,MOVER
        DATA FCTN4,PGMEM
        DATA FCTN5,SEARCH
        DATA FCTN6,PGMEM
        DATA FCTN8,MEMDMP
        DATA FCTN9,MEMEDT
        DATA 0
*
KEY2    DATA FCTN4,PAGEUP
        DATA LFBRAK,PGUPSF
        DATA FCTN5,SEARCH
        DATA FCTN6,PAGEDN
        DATA UPAROW,UPK
        DATA DNAROW,DWNK
        DATA FCTN8,MEMDMP
        DATA 0
*
KEY35   DATA RTAROW,RTK
        DATA LFAROW,LFK
        DATA FCTNEQ,ASCHQT
        DATA FCTN0,BSWTCH
        DATA 0

KEY3    DATA FCTN9,RTNFED
        DATA FCTN1,SWPMMM
        DATA FCTN2,MOVER
        DATA FCTN3,MOVER
        DATA FCTN7,COLOR
        DATA ENTER,HOME
        DATA 0
*
KEY4    DATA FCTN5,SRCHRT
        DATA FCTN9,DEFSTR
        DATA 0
*
KEY5    DATA ENTER,SRCMEM
        DATA FCTN5,SRCHRT
        DATA FCTN9,DEFRTN
```

DATA 0

```
SCLEN  EQU  >8355
SCNAME EQU  >8356
CRULST EQU  >83D0
SADDR  EQU  >83D2
*
***DATA
*
DECMAL TEXT '.'
HAA    BYTE >AA
*
***UTILITY BLWP VECTORS
*
DSRLNK DATA DLNKWS,DLENTR     Link to device service routine
*
***LINK TO DEVICE SERVICE ROUTINE
*
DLENTR SZCB @H2000,R15        Reset equal bit
       MOV  @SCNAME,R0        Fetch pointer into PAB
       MOV  R0,R9             Save pointer
       AI   R9,-8             Adjust pointer to flag byte
       BL   @VSBR             Read device name length
       MOVB R1,R3             Store it elsewhere
       SRL  R3,8              Make it a word value
       SETO R4                initialize a counter
       LI   R2,NAMBUF         Point to NAMBUF
LNK$LP INC  R0                Point to next char of name
       INC  R4                Increment character counter
       C    R4,R3             End of name?
       JEQ  LNK$LN            YES
       CB   *R2+,@DECMAL      Have we hit a decimal point?
       JNE  LNK$LP            NO
LNK$LN CI   R4,7              Is name length >7
       JGT  LNKERR            YES! Error
       CLR  @CRULST
       MOV  R4,@SCLEN-1       Store name length for search
       INC  R4                Adjust it
       A    R4,@SCNAME        Point to position after name
*
***SEARCH ROM FOR DSR
*
SROM   LWPI GPLWS             Use GPL workspace to search
       CLR  R1                Version found of DSR etc.
       LI   R12,>0F00         start over again
NOROM  MOV  R12,R12           Anything to turn off
       JEQ  NOOFF             NO
HB1E   EQU  $
       SBZ  0                 YES! Turn it off
H100   EQU  $+2
NOOFF  AI   R12,>0100         Next ROM's turn on
       CLR  @CRULST           Clear in case we're finished
       CI   R12,>2000         At the end
       JEQ  NODSR             No more ROMs to turn on
       MOV  R12,@CRULST       Save address of next CRU
HEDGE  EQU  $
       SBO  0                 Turn on ROM
       LI   R2,>4000          Start at beginning
       CB   *R2,@HAA          Is it a valid ROM?
       JNE  NOROM             NO
       AI   R2,8              Go to first pointer
       JMP  SG02
```

```
SGO     MOV  @SADDR,R2          Continue where we left off
        SBO  0                  Turn ROM back on
SGO2    MOV  *R2,R2             Is address a zero
        JEQ  NOROM              YES! No program to look at
        MOV  R2,@SADDR          Remember where we go next
        INCT R2                 Go to entry point
        MOV  *R2+,R9            Get entry address
*
***SEE IF NAME MATCHES
*
        MOVB @SCLEN,R5          Get length as counter
        JEQ  NAME2              Zero length, don't do match
        CB   R5,*R2+            Does length match?
        JNE  SGO                NO
        SRL  R5,8               Move to right place
        LI   R6,NAMBUF          Point to NAMBUF
NAME1   CB   *R6+,*R2+          Is character correct?
        JNE  SGO                NO
        DEC  R5                 More to look at?
        JNE  NAME1              YES
NAME2   INC  R1                 Next version found
        BL   *R9                Match, call subroutine
        JMP  SGO                Not right version
        SBZ  0                  Turn off ROM
        LWPI DLNKWS             Select DSRLNK workspace
        MOV  R9,R0              Point to flag byte in PAB
        BL   @VSBR              Read flag byte
        SRL  R1,13              Just want the error flags
        JNE  IOERR              ERROR!
        RTWP
*
***ERROR HANDLING
*
NODSR   LWPI DLNKWS             Select DSRLNK workspace
LNKERR  CLR  R1                 Clear the error flags
IOERR   SWPB R1
        MOVB R1,*R13            Store error flags in calling R0
        SOCB @H2000,R15         Indicate an error occured
        RTWP                    Return to caller
```

```
*===============================================================================
*
*       FIELD EDITOR (STACK DRIVEN)
*
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
* Entry to home cursor while on status screen
EDIT    LI    R1,FIELD          Move first field address location into R1
        SZC   @HC,R15           Set keyboard route for F-EDITOR
*
EDIT2   EQU   $
*
* Second entry. R1 must contain current field address and FRSTFD/LSTFLD must be
*  initialized with first and last field addresses in stack
EDIT1   MOV   R1,@FLDPNR        Move R1 into FLDPNR
        MOV   *R1,R12           Move first field address into R12
        ANDI  R12,>3FF          Mask off info bits
        CLR   R5                Clear field position pointer
EDIT3   BL    @CRINIT           Set up cursor
POPSTK  MOV   @FLDPNR,R2        Move field stack pointer into R2
        MOV   *R2+,R6           Move address word into R6
        SLA   R6,4              Mask off first four bits
        SRL   R6,14             Mask off last ten bits and leave field length
        MOV   *R2+,@VALPNR      Move validation stack pointer into VALPNR
        MOV   *R2,@SOPPNR       Move special operations vector into SOPPNR
*-------------------------------------------------------------------------------
* Keyboard scan routine
*
KBSCAN  BL    @CURSOR           Service cursor
        BL    @CHKKEY           Scan keyboard
        COC   @H2000,R0         Is the condition bit set?
        JEQ   KBSCA2            YES!
        CB    @HBFF,R1          Is KEY a null?
        JNE   KBSCA1            NO!
        MOV   @DCOUNT,@DELAY    Restore DELAY counter
        JMP   KBSCAN            Loop
KBSCA1  MOV   @DELAY,@DELAY     Is the auto-repeat timer zero?
        JEQ   KBSCA2            YES!
        DEC   @DELAY            Decrement DELAY before beginning auto-repeat
        JMP   KBSCAN            Loop if not ready for auto-repeat yet
KBSCA2  MOV   @SPEED,R0         Move SPEED into R0
KBSCA3  DEC   R0                This loop slows auto-repeat cursor movement
*                                down to the human domain!!
        JGT   KBSCA3            If not >FFFF then loop
        CZC   @HC,R15           Is key route to F-EDITOR?
        JNE   MEKEY             NO!
*-------------------------------------------------------------------------------
* Service F-EDITOR keyboard

FEKEY   LI    R2,SPDSET         Load normal scroll speed in case
        MOV   R2,@SPEED          we just did a memory page up/down
        LI    R2,KEY14          Load KEY14 stack address
        BL    @KEYVEC           Look for a key match
        LI    R2,KEY1           Load KEY1 stack address
        BL    @KEYVEC           look for a key match
        B     @ENTRY            No match so check for data entry
*-------------------------------------------------------------------------------
* Service MEDIT
*
MEKEY   COC   @HC,R15           Check key route
        JNE   SRHKEY            Jump if not for MEDIT
```

```
        BL    @SCANCT           Grab column 0 keyboard scan
        CLR   @SPEED            Go to full speed for scrolling and paging
        LI    R2,KEY2           Load KEY3 stack address
        BL    @KEYVEC           Look for a key match
MEKEY1  LI    R2,SPDSET         Return to normal speed
        MOV   R2,@SPEED           .
        LI    R2,KEY35          Load KEY35 stack address
        BL    @KEYVEC           Look for a key match
        LI    R2,KEY3           Load KEY 3 stack address
        BL    @KEYVEC           Look for a key match
        B     @MEDIT            No match so check for data entry
*
*--------------------------------------------------------------------------
* Service search routine
*
SRHKEY  LI    R2,SPDSET         Load normal cursor repeat speed
        MOV   R2,@SPEED           .
        COC   @H10,R15          Are we in entering a string?
        JEQ   SRHKE1            YES!
        LI    R2,KEY14          Load KEY14 stack address
        BL    @KEYVEC           Look for a key match
        LI    R2,KEY4           Load KEY4 stack address
        BL    @KEYVEC           Look for a key match
        B     @ENTRY            Validate an entry
SRHKE1  SETO  R6                Clean out keyboard column 0 register
        LI    R2,KEY35          Load KEY35 stack address
        BL    @KEYVEC           Look for a key match
        LI    R2,KEY5           Load KEY5 stack address
        BL    @KEYVEC           Look for a key match
        B     @MEDIT            Validate an entry
*
*--------------------------------------------------------------------------
* Enter data after validation
*
ENTRY   MOV   @VALPNR,R3        Move VALIDATE stack pointer into R3
        MOVB  *R3+,R4           Move TYPE byte into R4
        MOV   @SOPPNR,@SPCL01   Move SPECIAL OPERATIONS link into SPCL01
        JNE   VALDTE            If it is non-zero then jump
        LI    R2,SPCL02         Move SPCL02 address into SPCL01
        MOV   R2,@SPCL01          .
*
VALDTE  CB    *R3,@HBFF         Are we at the EOStack of validation?
        JEQ   SPCLOP            YES!
        CB    *R3+,R1           Check entry against stack values
        JNE   VALDTE            Loop until match or EOStack
        DEC   R3                Set condition bit for special operation and
*                                move back up one in the stack in case we
*                                were on the last comparison value
*
*--------------------------------------------------------------------------
* Special operation definition. NOTE: You must not destroy
*  R1, R3, R4 or R12
*
SPCL01  EQU   $+2
SPCLOP  BL    @SPCL02           Perform special operation if any. DYNAMIC operand
SPCL02  CB    *R3,@HBFF         Was there a match in validation?
        JNE   SPCL04            YES!
SPCERR  EQU   $                 Move error screen colors into R0
        BL    @ERRCOL           Set screen up with new colors
        B     @POPSTK           Return through POPSTK
```

```
SPCL04 MOV   @FLDPNR,R3        Move field stack pointer into R3
       AI    R3,6              Adjust R3 to point to DATA value
*
*--------------------------------------------------------------------------------
* Check for numeric entry
*
NUMENT SLA   R4,1              Are we handling numerics?
       JNC   UCASE             NO!
       MOVB  R1,@SCRVLU        Move the key value into SCRVLU for CURSOR
       CI    R1,>4000          Is it alpha?
       JL    NUMEN1            NO!
       ANDI  R1,>5F00          Make sure it's upper case
       MOVB  R1,@SCRVLU        Move new character into CURSOR's SCRVLU
       AI    R1,->700          Remove alpha offset
NUMEN1 AI    R1,->3000         Remove ASCII offset
       JNE   NUMEN2            If non-zero then jump
*                               entries of zero to advance to subsequent fields
NUMEN2 SLA   R1,4              Align nybble
       MOV   R5,R0             Move present field position into R0
       SLA   R0,2              Multiply R0 by 4
       LI    R2,>F000          Load nybble mask into R2
       SRC   R1,0              Shift new value to field position (nybble wise)
       SRC   R2,0              Shif mask to field position (nybble wise)
       SZC   R2,*R3            Remove nybble from DATA value
       A     R1,*R3            Put new nybble into DATA value
       JMP   ALPEN1            Advance cursor
*
*
*--------------------------------------------------------------------------------
* Convert lower case alphas to upper case
*
UCASE  SLA   R4,1              Are we handling upper case only?
       JNC   LCASE             NO!
       CI    R1,>7A00          Check for 'z' boundary
       JH    ALPENT            If high then skip
       CI    R1,>6100          Check for 'a' boundary
       JL    ALPENT            If low then skip
       ANDI  R1,>5F00          Mack lower case to upper case
       JMP   ALPENT            Put it into DATA value
*
*--------------------------------------------------------------------------------
* Convert upper case alphas to lower case
*
LCASE  SLA   R4,1              Are we handling lower case only?
       JNC   ALPENT            NO!
       CI    R1,>5A00          Check for 'Z' boundary
       JH    ALPENT            Skip if high
       CI    R1,>4100          Check for 'A' boundary
       JL    ALPENT            Skip if low
       ORI   R1,>2000          Mask upper case to lower case
*
*--------------------------------------------------------------------------------
* Alpha entry
*
ALPENT A     R5,R3             Add field position (0 or 1) to DATA pointer
       MOVB  R1,*R3            Move alpha into DATA
       MOVB  R1,@SCRVLU        Move new alpha into CURSOR's SCRVLU
ALPEN1 BL    @CROFF            Turn cursor off to display new value immediately
       CLR   R1                Clear R1 so RTKEY doesn't think it's a RTAROW
       JMP   RTKEY             Advance cursor
```

```
*
*--------------------------------------------------------------------------------
*
*
POPRTN MOV  *R2,R12          Move new address into R12
POPRT1 ANDI R12,>3FF         Mask off info bits
       MOV  R2,@FLDPNR       YES! Move new field stack pointer into FLDPNR
POPVEC EQU  $+2
POPRT2 B    @POPSTK          Set up next field
KRTVEC EQU  $+2
KRTN   B    @KBSCAN          Branch to KBSCAN
*
*--------------------------------------------------------------------------------
* Left arrow key
*
LFKEY  MOV  R5,R5            Are we at the BOField?
       JEQ  LFKEY1           YES!
       DEC  R5               Decrement field pointer
       DEC  R12              Decrement cursor position
       JMP  KRTN             Return to KEYS
LFKEY1 MOV  @FLDPNR,R2       Move current field pointer into R2
LFKEY2 AI   R2,-8            Point to previous field stack address
LFKEY3 MOV  *R2,R3           Grab previous field
       JNE  LFKEY4           If not at the stack beginning keep moving
*                             left
       LI   R2,LSTFLD        Load R2 with last field if at BOField
       JMP  LFKEY3           Go wrap around to bottom of screen
LFKEY4 SLA  R3,1             Should we skip this field?
       JOC  LFKEY2           YES!
       MOV  *R2,R12          Move address into R12
       SLA  R12,4            Mask off first nybble
       SRL  R12,14           Clear rest of address and leav field length
       MOV  R12,R5           Update field position register
       A    *R2,R12          Add address to field length
       JMP  POPRT1           Set up for return
*
*--------------------------------------------------------------------------------
* Right arrow key
*
RTKEY  C    R5,R6            Are we at the EOField?
       JEQ  RTKEY1           YES!
       INC  R5               Bump field pointer
       INC  R12              Increment cursor position
       JMP  KRTN             Return to KEYS
RTKEY1 MOV  @FLDPNR,R2       Move current field pointer address into R2
RTKEY2 AI   R2,8             Point to next field
RTKEY3 MOV  *R2,R3           Move new address into R3
       JNE  RTKEY4           NO!
       LI   R2,FIELD         YES! Load top of field into R2
       JMP  RTKEY3           Wrap around to top left of screen
RTKEY4 SLA  R3,1             Should we skip this field?
       JOC  RTKEY2           YES!
       CLR  R5               Update field position register
       JMP  POPRTN           Set up for return
* Page memory block up or down from status block
PGMEM  CLR  @SPEED           Go to high speed
       MOV  R1,R2            Save key press in R2
       BL   @CROFF           Turn the cursor off
       MOV  @MADDR,R14       Force R14 to the same value as MADDR
       MOV  @LR3VEC,@TEMP2◊  Save LR3VEC
```

```
        LI    R0,PGMEM2          Return from LR3
        MOV   R0,@LR3VEC         Load return into LR3VEC
        CI    R2,FCTN6           Are we paging down?
        JNE   PGMEM1             NO!
        B     @PAGEDN            YES!
PGMEM1  B     @PAGEUP            Page up
PGMEM2  MOV   @TEMP20,@LR3VEC    Restore LR3VEC
        B     @EDIT3             Return through EDIT3
```

```
*
*===============================================================
*
*       MEMORY EDIT INTIALIZATION SECTION
*
*---------------------------------------------------------------
*
MEMEDT  BL    @CROFF              Turn cursor wherever it may be
MEMED1  MOV   @TEMP3,R12          Move starting screen address into R12 for CURSOR
        A     @MADDR,@TEMP2       Add MADDR to cursor difference within window
        CB    @HV,@CGV            Are we in VDP?
        JNE   MEMED2              NO!
        SZC   @HC000,@TEMP2       Mask new cursor address
MEMED2  MOV   @TEMP2,R14          Add cursor to memory block through R14
        MOV   R14,R0              Write MADDR out to screen
        BL    @BIHEXW                .
        LI    R0,TOP+43              .
        LI    R1,HEXW$               .
        LI    R2,4                   .
        BL    @VMBW                  .
        SOC   @HC,R15             Set key route to memory block editor
MEMED3  BL    @CRINIT             Put cursor up in the home position
        JMP   MEDIT2              Branch to KBSCAN
*
*===============================================================
*
*       MEMORY BLOCK EDITOR
*
*---------------------------------------------------------------
*
MEDIT   COC   @H100,R15           Is .A set for ASCII?
        JEQ   MEDIT3              YES!
*---------------------------------------------------------------
* Validate key for hex entry
*
        LI    R3,HEXSTK+1         Load address of HEX validation stack
MEDIT1  CB    *R3,@HBFF           Are we at the EOStack?
        JEQ   MEDERR              YES! Return with no match
        CB    *R3+,R1             Do we have a match?
        JNE   MEDIT1              NO!
        CI    R1,>3A00            Do we have a numeral?
        JL    MEDIT4              YES!
        ANDI  R1,>5700            Make sure that alpha is upper case
        JMP   MEDIT4
MEDERR  EQU   $
        BL    @ERRCOL             Set screen up with new colors
MEDVEC  EQU   $+2
MEDIT2  B     @KBSCAN             Return vector to KBSCAN
*---------------------------------------------------------------
* Validate key for alpha entry
*
MEDIT3  CI    R1,>2000            Is alpha less than a space character?
        JL    MEDERR              YES! Return
HB7E    EQU   $+2
        CI    R1,>7E00            Is alpha greater than a '~' character?
        JH    MEDERR              YES! Return
*---------------------------------------------------------------
* Put new character on the screen
*
MEDIT4  MOV   R1,@SCRVLU          Move key value into SCRVLU for CURSOR
```

```
        COC   @H100,R15         Is .A set for ASCII?
        JNE   MEDIT9            NO!
        COC   @H1000,R15        Is the basic offset switch set?
        JNE   MEDIT7            NO!
        AI    R1,-OFFSET        Add >60 to entry
        JMP   MEDIT7            Put value into memory
*--------------------------------------------------------------------------
* Reduce new hex entry to binary
*
MEDIT9  BL    @CROFF            Turn cursor momentarily
        MOV   R12,R0            Restore address into R0
MEDI12  LI    R1,TEMP1          Load R1 with address of TEMP1
        LI    R2,2              Two bytes to read from screen
        BL    @VMBR             Fetch two HEX characters from screen
        CB    @HEDGE,@TEMP1+1   Was the MSCharacter an EDGE character?
        JNE   MEDI13            NO!
        DEC   R0                YES! Back up one and try again (This way we
*                                don't have to keep track of the cursor!)
        JMP   MEDI12
MEDI13  MOV   @TEMP1,R3         Move new hex value into R3
        AI    R3,->3030         Subtract ASCII offset from both bytes
        CB    R3,@HBA           Is MSB alpha?
        JL    MEDIT5            NO!
        AI    R3,->700          Subtract alpha offset from MSB
MEDIT5  MOV   R3,@TEMP1         Move value into TEMP1
        CB    @TEMP1+1,@HBA     Is LSB alpha?
        JL    MEDIT6            NO!
        AI    R3,->7            Subtract alpha offset from LSB
*--------------------------------------------------------------------------
* Put new binary hex entry into MSB of R1

MEDIT6  CLR   R1                Clear register R1
        SLA   R3,4              Align high nybble
        SOC   R3,R1             Put both nybbles into R1
        SLA   R3,4              Align low nybble
        SOC   R3,R1             Put second nybble into R1. We now have our binary
*                                number in the MSB of R1. The garbage in the LSB
*                                of R1 does not concern us
*--------------------------------------------------------------------------
* Write new byte out to appropriate memory
*
MEDIT7  CB    @HG,@CGV          Is it a GRAM operation?
        JNE   MEDIT8            NO!
        MOV   R14,R0            Set up R0 with GRAM address
        BL    @GWADD            Set up GRAM address
        BL    @GWDAT            Write data in R1 to GRAM
        JMP   MEDI11
MEDIT8  CB    @HV,@CGV          Is it a VDP operation?
        JNE   MEDI10            NO!
        MOV   R14,R0            Load VDP address
        BL    @VSBW             Write byte
        JMP   MEDI11
MEDI10  BL    @GWADD            Make sure we are in the proper library slot
        MOVB  R1,*R14           Put byte into CPU RAM if none of the above
MEDI11  BL    @MEMBLK           Verify memory write. This was added for GROM's
*                                sake. (Since you can't write to GROM!!)
        BL    @CRINIT           Put the cursor back up on the screen
MEDI14  CLR   R1                Disable the horizontal scroll in RTK in case
*                                we're at the EOLine
        JMP   RTK               Advance cursor and return to KBSCAN
```

```
*
*===============================================================================
*
*      UP/DOWN ARROW KEYS FOR MEMORY BLOCK EDIT
*
*-------------------------------------------------------------------------------
* Entry for up arrow key
*
UPK     LI    R4,-40              Initialize variables
        LI    R13,-12                .
        JMP   UD
*
*-------------------------------------------------------------------------------
* Entry for down arrow key
*
DWNK    LI    R4,40               Initialize variables
        LI    R13,12                 .
*-------------------------------------------------------------------------------
* Common code for both up and down arrow keys
*
UD      MOV   R12,R0              Move cursor position into R0
        A     R4,R0               Add/Subtract 12 to/from R0
        C     R0,@BLKSCN          Are we ready to scroll down?
        JL    UD6                 YES!
        BL    @VSBR               Read a byte a line ahead
        C     R1,@HEDGE           Are we ready to scroll up? (First line below the
*                                  memory block must contain EDGE, SLINE or DLINE
*                                  characters for this to work!!)
        JLE   UD6                 YES!
        LI    R1,SPDSET           NO! Slow auto-repeat down to normal level
        MOV   R1,@SPEED              .
*-------------------------------------------------------------------------------
* Move the cursor up a line
*
        A     R4,R12              Adjust cursor position
* Check for shift key to freeze cursor
        CZC   @H2000,R6           Is the shift key depressed?
        JNE   UD3                 NO!
UD5     MOV   @MADDR,R2           YES! Move MADDR into R2
        S     R13,R2              Subtract 12 from it
        CLR   R13                 Clear R13 so cursor isn't advanced any
        JMP   UD4                 Move block up a line and cursor with it
*-------------------------------------------------------------------------------
* Scroll the memory block up or down a line
*
UD6     COC   @H10,R15            Is SEARCH using us?
        JEQ   RTK1                YES! Don't scroll
        CZC   @H2000,R6           Is the shift key depressed?
        JEQ   RTK1                YES! Stop scrolling, we're at the top/bottom
UD1     MOV   @MADDR,R2           Move block address into R2
        A     R13,R2              Add/Subtract 12 to/from R2
UD4     CB    @HV,@CGV            Are we working with VDP memory?
        JNE   UD2                 NO!
        ANDI  R2,>3FFF            Make sure VDP RAM wraps around on 16K boundary
UD2     MOV   R2,@MADDR           Restore new memory block address
        BL    @MEMBLK             Scroll memory block
        BL    @CRINIT             Restore cursor to screen after moving memory
UD3     A     R13,R14             Adjust cursor memory pointer
        JMP   LR2                 Put new address on screen
*
```

```
*===============================================================================
*
*       RIGHT/LEFT ARROW KEY ROUTINE FOR THE MEMORY BLOCK EDITOR
*
*-------------------------------------------------------------------------------
* Entry for right arrow key
*
RTK     LI    R13,1               Add one to R13
        MOV   @MADDR,R3           Move current memory address into R3
        A     @BYTENO,R3          Add the total number of bytes on screen to R3
        DEC   R3                  Adjust R3 by one
        CB    @HV,@CGV            Are we in VDP?
        JNE   RTK3                NO!
        ANDI  R3,>3FFF            Mask address
RTK3    C     R14,R3              Are we at the last character in the block?
        JNE   RTK2                NO!
        MOV   R1,R4               Move last key press into R4 for safe keeping
        MOV   R12,R0              Move cursor address into R0
        INC   R0                  Bump address by one
        BL    @VSBR               Read character at that address
        CB    R1,@HEDGE           Is it an EDGE character?
        JNE   RTK2                NO!
        CI    R4,RTAROW           Check to see if last key pressed was a RTAROW ke
        JEQ   UD6                 NO! Don't do a horizontal scroll
RTKRTN  EQU   $+2
RTK1    B     @KBSCAN             Return to through LR2
H1      EQU   $+2
RTK2    LI    R2,1                Load variables
        JMP   LR                  Jump to main routine
*
*-------------------------------------------------------------------------------
* Entry for left arrow key
*
LFK     LI    R13,-1              Load R13 with -1 for a horizontal scroll
        C     R12,@BLKSCN         Are we at the BOL?
        JEQ   UD6                 YES! Do a horizontal scroll
        LI    R2,-1               Load R2 to advance cursor one to left
*-------------------------------------------------------------------------------
* Common code shared by both right and left arrow keys
* Deal with HEX screen
*
LR      A     R2,R12              Ad/Devance the cursor
        MOV   R12,R4              Save address in R4
        BL    @CURSOR             Make the change
        CZC   @H2000,R6           Is the shift key depressed?
        JNE   LR4                 NO!
        COC   @H100,R15           YES! Is ASCII on the screen?
        JEQ   UD5                 YES! Freeze cursor
        C     R12,R4              Compare present cursor address with previous add
        JEQ   RTK1                No change so we must be on the same byte
        A     R2,R12              We'll just passed over to the next byte
        JMP   UD5                 Do left/right move with cursor frozen
LR4     C     R12,R4              Did we just encounter an EDGE character?
        JEQ   LR2                 NO! Don't advance address pointer R14
*-------------------------------------------------------------------------------
* Deal with ASCII screen
*
LR1     A     R2,R14              Adjust cursor address pointer by one either way
LR2     COC   @H10,R15            Is SEARCH using us?
        JEQ   RTK1                YES!
```

```
        CB    @HV,@CGV          Are we servicing VDP?
        JNE   LR3               NO!
        ANDI  R14,>3FFF         Make sure we wrap around 16K boundary
*-------------------------------------------------------------------------------
* Put new address on screen
*
LR3     MOV   R14,R0            Move cursor address pointer into R1
        BL    @BIHEXW           Get ASCII representation of address
        LI    R0,TOP+43         Load address of MADDR
        LI    R1,HEXW$          Load HEX$ address
        LI    R2,4              Four bytes to write
        BL    @VMBW             Put new address on screen
LR3VEC  EQU   $+2
        B     @RTK1             Return to KBSCAN
*-------------------------------------------------------------------------------
* Page up and down routine
*
PGUPSF  CZC   @H2000,R6         These next three lines fix a bug. Before if you
        JEQ   PAGEUP             paged up with the shift key depressed, you would
        B     @MEKEY1            end up with '{'s on the screen
PAGEUP  MOV   @BYTENO,R3        Move byte count into R3
        JMP   PAGE
PAGEDN  MOV   @BYTENO,R3        Move byte count into R3
        NEG   R3                Negate byte count
PAGE    A     R3,@MADDR         Add byte count to MADDR
        A     R3,R14            Add byte count to cursor address pointer
        CB    @HV,@CGV          Are working with VDP?
        JNE   PAGE1             NO!
        LI    R4,>C000          Load wrap around mask
        SZC   R4,@MADDR         Mask MADDR for VDP wrap around
        SZC   R4,R14            Mask cursor address pointer for VDP wrap around
PAGE1   BL    @MEMBLK           Page memory block
        CZC   @HC,R15           Are we in the STATUS block?
        JEQ   PAGE2             YES!
        BL    @CRINIT           Restore cursor to screen
PAGE2   JMP   LR2               Return to KBSCAN
*-------------------------------------------------------------------------------
* Routine to re-enter F-EDITOR from MEDIT
*
RTNFED  MOV   R14,@TEMP2        Save cursor memory address
        MOV   R12,@TEMP3        Save cursor screen address
        S     @MADDR,@TEMP2     Calculate cursor memory address-MADDR diff
        MOV   @MADDR,R14        Restore cursor address pointer to start of block
        CB    @HV,@CGV          Are we in VDP?
        JNE   RTNFE1            NO!
        SZC   @HC000,@TEMP2     YES! Mask diff
RTNFE1  LI    R0,RTNFE2         Load R0 with RTNFE1
        MOV   R0,@RTKRTN        Load RTKRTN with RTNFE1 vector to return to this
*                                routine when LR is done
        JMP   LR2               Update address on screen
RTNFE2  BL    @CROFF            Return here and turn cursor off
        SZC   @HC,R15           Set key route for F-EDITOR
        MOV   @MEDVEC,@RTKRTN   Restore FLDPNR vector with KBSCAN
        MOV   @FLDPNR,R1        Restore FLDPNR to R1 in preparation for returnin
        B     @EDIT1            Enter F-EDITOR at previous field
*
*-------------------------------------------------------------------------------
* HOME CURSOR
*
HOME    CLR   @TEMP2            Clear diff
```

```
LI    R0,TOP+161        Load home position
MOV   R0,@TEMP3         Reset cursor screen address
B     @MEMEDT           Re-initialize MEDIT
```

```
*
*===============================================================================
*
*       MEMORY BLOCK WRITE
*
*-------------------------------------------------------------------------------
*
BLKSCN  DATA TOP+161            Starting screen location of memory block. DYNAMIC
BYTENO  DATA 144                Number of bytes to write to screen. Also DYNAMIC!
*-------------------------------------------------------------------------------
* Some initialization
*
MEMBLK  MOV  R11,R9             Save return address
        BL   @GRDAT             Perform a dummy GROM read. This is to set the
*                                ROM slot for a cartridge library.
        MOV  @MADDR,R7          Move memory address into R7
        MOV  @BLKSCN,R3         Move starting screen address into R3
        MOV  @BYTENO,R6         Move number of bytes to transfer into R6
*-------------------------------------------------------------------------------
* Start of two loops. MEMBL1 is outer and MEMB10 is inner
* Determine from what type of memory we are getting bytes
*
MEMBL1  CLR  R4                 Initialize OP$ pointer
MEMB10  DEC  R6                 Decrement byte count
        JLT  MEMB11             If done, then return to caller
        BL   @FETCH             Grab byte from memory
*-------------------------------------------------------------------------------
* Determine if it is HEX or ASCII data on the screen
*
MEMBL5  COC  @H100,R15          Is the ASCII flag on?
        JEQ  MEMBL6             YES!
        MOVB R1,R0              Move value into R0 for BIHEX
        BL   @BIHEXB            Get ASCII HEX representation of byte
        MOV  @HEXB$,R1          Load R1 with address of HEX$
        JMP  MEMBL9
*-------------------------------------------------------------------------------
* Check range on ASCII entry
*
MEMBL6  COC  @H1000,R15         Is the basic offset switch on?
        JNE  MEMBL3             NO!
        AI   R1,OFFSET          Subtract >60 from data
MEMBL3  CB   R1,@HB20           Is alpha lower than space character?
        JL   MEMBL7             YES!
        CB   R1,@HB7E           Is alpha higher than '~' character?
        JLE  MEMBL8             YES!
MEMBL7  MOVB @HB1E,R1           Move TIC into place of non-printable character
MEMBL8  SWPB R1                 Prepare to write LSB
        MOVB @HEDGE,R1          Move an EDGE into LSB of R1
        SWPB R1                 Restore R1
*-------------------------------------------------------------------------------
* Build OP$. Transferring a whole line from CPU to VDP screen area makes the
*  routine a little longer but it makes the scroll much cleaner (no waviness)
*
MEMBL9  MOVB R1,@OP$(R4)        Move MSB byte into OP$
        SWPB R1                 Move LSB into place
        MOVB R1,@OP$+1(R4)      Move LSB byte into OP$
        AI   R4,3               Increment OP$ pointer
        CI   R4,36              Have we built an entire line?
        JNE  MEMB10             NO! Keep stuffing
*-------------------------------------------------------------------------------
```

```
* Put a line on the screen
*
        MOV   R3,R0             Load screen address into R0
        AI    R3,40             Increment screen address to next line
        LI    R1,OP$            Load address of OP$ into R1
        LI    R2,36             Thirty-four bytes to write
        BL    @VMBW             Put OP$ on the screen
        JMP   MEMBL1
MEMB11  SZC   @H2,R15           Reset byte/word flag
        B     *R9               Return to caller
```

```
*
*--------------------------------------------------------------------------------
* CLOSE FILE
*
CLS$    DATA >0112              Close file info for PAB
CLOSE   MOV  R11,R10            Save return
        LI   R0,>F80            Address of PAB
        LI   R1,CLS$            Close file info
H2      EQU  $+2                >2
        LI   R2,2               Two bytes to write
        BL   @VMBW              Set PAB for close
* Close file
        MOV  @TEMP20,@SCNAME    Restore PAB name pointer
        BLWP @DSRLNK            Close file
*
        B    *R10               Return
*
*--------------------------------------------------------------------------------
* ERROR REPORTING
*
ERROR   JNE  ERROR1             No error to report
        BL   @BIHEXB            Get hex of error
        LI   R0,TOP+914         Set up to put
        LI   R1,HEXB$            error on the screen
        LI   R2,2                  .
        BL   @VMBW                 .
*
MEMDN   BL   @CLOSE             Close file
*
ERRTN   BL   @CHKKEY            Dummy key scan
        CB   @HBFF,R1           Is a key still down
        JNE  ERRTN              YES!
        MOV  @WSR+30,R15        Grab main program flags
        BL   @MEMBLK            Rewrite memory block
        LWPI WSR                Load main workspace
        BL   @CRINIT            Put up cursor
        B    @POPSTK            Return to main program
*
ERROR1  RT                      Return to caller
*
*--------------------------------------------------------------------------------
* OPEN FILE
*
MEMDMP  BL   @CROFF             Turn cursor off
        LWPI UTILWS             Load UTILITY workspace
        MOV  @AOP,R7            Grab start address
        C    @AOP,@BOP          Is finish less than start
        JHE  ERRTN              YES!
        MOV  @BOP,R9            Grab finish
        S    @AOP,R9            Calculate number of bytes to write
        INC  R9                    .
OPEN    LI   R0,>F80            Load PAB into VDP RAM
        LI   R1,PAB                .
        LI   R2,9                  .
        BL   @VMBW                 .
* Get device name from memory
        LI   R0,DNAME           Load address of filename
        LI   R1,80              Max. of 80 characters in filename
        CLR  R2                 Initialize counter
OPEN1   CB   *R0+,@HB20         Parse for a space
```

```
        JEQ   OPEN2               Found end so get out
        INC   R2                  Increment name length counter
        DEC   R1                  Decrement max. length counter
        JNE   OPEN1               Loop if haven't reached 80 chars
* Put device name and length into PAB
OPEN2   SWPB  R2                  Move name length into position
        MOVB  R2,@DNAME-1         Move name length just before name
        JEQ   ERRTN               If zero in length then indicate an error
        SWPB  R2                  Restore name length
        INC   R2                  Add one to it to include name and length byte
        LI    R0,>F80+9           Move length byte and name into VDP
        MOV   R0,@TEMP20             Save length pointer in TEMP20
        LI    R1,DNAME-1              .
        BL    @VMBW                   .
* Open file
        MOV   @TEMP20,@SCNAME     Open file
        BLWP  @DSRLNK                .
* Go check for an error
        BL    @ERROR              Check for an error
* No error so set PAB for writing
        LI    R0,>F80             Set PAB for write
        LI    R1,>0300                .
        BL    @VSBW                   .
* Put 'ok' message up on screen
        LI    R0,TOP+914          Indicate 'ok' on screen
        LI    R1,OK                   .
        LI    R2,2                    .
        BL    @VMBW                   .
*
*--------------------------------------------------------------------------------
* MEMORY DUMP
*
*
MEMD11  MOV   R9,R9               Are we done?
        JEQ   MEMDN               YES!
        LI    R3,12               Load 12 bytes per record
        C     R9,R3               Less than 12 bytes left?
        JHE   MEMDM5              NO!
        MOV   R3,R9               Move 12 into R3
MEMDM5  S     R3,R9               Subtract 12 bytes from total byte count
*
H1000   EQU   $+2                 >1000
        LI    R6,>1000            Load PAB buffer address in VDP
*
        LI    R1,'> '             Put a '>' into string
        BL    @STUFF                  .
*
        MOV   R7,R0               Move address into R0
        BL    @BIHEXW             Get hex of address
        LI    R4,-4               Initialize a counter
MEMDM1  MOVB  @HEXW$+4(R4),R1     Move address into string
        BL    @STUFF                  .
        INC   R4                      .
        JLT   MEMDM1                  .
*
        MOV   R3,R4               Twelve bytes to stuff
        MOV   R7,R5               Save address into R5
MEMDM2  MOVB  @HB20,R1            Put a space into string
        BL    @STUFF                  .
*
```

```
        BL   @FETCH             Grab a byte from memory
        BL   @BIHEXB            Convert into hex
        MOV  R6,R0              Move PAB buffer address into R0
        INCT R6                 Increment address pointer
        LI   R1,HEXB$           Put hex into buffer
        LI   R2,2                  .
        BL   @VMBW                 .
        DEC  R4                 Decrement 12 byte counter
        JNE  MEMDM2             Loop until done
* Initialize registers for twice through string loop
        LI   R12,2              Go twice through loop 1-ASCII 2-ASCII w/BASIC BIA
        CLR  R13                Clear bias mask
* Restore R7  Put a space and ' into string
MEMDM6  MOV  R3,R4              Restore 12 byte count
        MOV  R5,R7              Restore start address
        MOV  @HB20,R1           Put a space into string
        BL   @STUFF                .
        LI   R1,'' '            Put a '' into string
        BL   @STUFF                .
* Put text into string
MEMDM3  BL   @FETCH             Grab a byte from memory
        AB   R13,R1             Add BASIC offset if any
        LI   R0,'* '            Assume an invalid char
        CB   R1,@HB20           Is it lower than a space?
        JL   MEMDM4             YES!
        CB   R1,@HB7E           Is it higher than a '~' ?
        JH   MEMDM4             YES!
        MOVB R1,R0              Get valid char
MEMDM4  MOVB R0,R1              Move char into R1
        BL   @STUFF             Put it into the string
        DEC  R4                 Loop until all 12 chars are in place
        JNE  MEMDM3                .
* Put the final ' into string
        LI   R1,>2700 '         Put a ''' into string
        BL   @STUFF                .
* If not done then setup for BASIC offset ASCII dump
        LI   R13,OFFSET         Load BASIC BIAS
        DEC  R12                Have we written BASIC BIAS chars yet?
        JNE  MEMDM6             NO!
* Write line out to device
        MOV  @TEMP20,@SCNAME    Write string out to device
        BLWP @DSRLNK               .
* Go check for an error
        BL   @ERROR             Go report any errors
* Do it all again
        B    @MEMD11            Loop until finished
*
*--------------------------------------------------------------------------------
* STUFF-Subroutine to stuff chars into VDP PAB buffer string
*
STUFF   MOV  R11,R10            Save return
        MOV  R6,R0              Grab PAB buffer address
        INC  R6                 Increment PAB buffer address for next access
        BL   @VSBW              Write byte to string
        B    *R10               Return
*
*--------------------------------------------------------------------------------
* FETCH ROUTINE. ENTER: Address in R7, automatically incremented
*
FETCH   MOV  R11,R10            Save return
```

```
        MOV   R7,R0           Put memory address into R0
        INC   R7              Increment memory address
        CB    @HV,@CGV        VDP?
        JNE   FETCH1          NO!
        ANDI  R0,>3FFF        Mask address
        SZC   @HC000,R7       Mask address
        BL    @VSBR           Get byte
        JMP   FETCH3          Get out
FETCH1  CB    @HG,@CGV        GROM?
        JNE   FETCH2          NO!
        BL    @GWADD          Set GROM address
        BL    @GRDAT          Get GROM byte
        JMP   FETCH3          Get out
FETCH2  MOVB  *R0,R1          CPU! Get byte
FETCH3  MOVB  R1,R0           Copy byte into R0 also
        B     *R10            Return
```

```
*--------------------------------------------------------------*
* MOVER - A Universal Move Memory Routine
*
*--------------------------------------------------------------*
*
MOVER   BL    @CROFF            Turn cursor off
        LWPI  UTILWS            Load UTILITY workspace
        MOV   @WSR+30,R15       Grab flags from main program
        MOV   @>83FA,R12        Grab GROM base from R13 of GPLWS
        MOV   @AOP,R0           Source
        MOV   @DOP,R1           Destination
        MOV   @BOP,R2           End of move
        C     R0,R2             Is destination less than source
        JHE   MOVER4            YES! Don't do routine
        S     R0,R2             Calculate number of bytes to move/fill
        INC   R2                Adjust
        LI    R6,>D554          Opcode for MOVB *R4,*R5

        CB    @KEYBRD,@H700     Is it a FILL
        JNE   MOVER2            No, continue on with Move
        MOV   R0,R1             Source becomes destination
        LI    R4,FOP            Yes, set up pointer to Fill byte
        CB    @CGV,@HV          Are we working with VDP?
        JEQ   TOV               YES!
        CB    @CGV,@HG          Are we working with GROM?
        JEQ   TOG               YES!
        JMP   TOC               Working with CPU

MOVER2  CB    @CGV,@HG          Is Source GROM
        JNE   FROMV             No, check VDP
        MOV   R12,R4            Yes, load R4 with Grom Read Data Address
        BL    @GSRCAD           Set Grom/Gram Source address
        CB    @DCGV,@HV         Is Destination VDP
        JEQ   TOV               Yes, the Destination is VDP
        CB    @DCGV,@HG         Are we going to GROM?
        JNE   TOC               No, Destination is CPU Ram
        MOV   R12,R5            No, set R5 with Grom Write data address
        AI    R5,>400             .
        JMP   GTOG1             Its a Grom/Gram to Gram move

FROMV   CB    @CGV,@HV          Is Source VDP
        JNE   FROMC             No, check CPU
        LI    R4,VDPRD            Yes, load R4 with VDP Read Data Address
        BL    @VRDADD           Set VDP Source address
        INC   R0                Increment R0 to next address
        CB    @DCGV,@HG         Is Destination GRAM
        JEQ   TOG               YES, Destination is Gram
        CB    @DCGV,@HV         Is Destination VDP?
        JNE   TOC               No, Destination is CPU Ram
        LI    R5,VDPWD            Yes, Set up R5 with VDP Write data address
H4000   EQU   $+2
        ORI   R1,>4000          Set up VDP Destination address for a write
        JMP   VTOV1             And goto VDP to VDP move

FROMC   MOV   R0,R4             CPU Source so set up R4 with Cpu address
        AI    R6,>20            Change R6 opcode to MOVB *R4+,*R5
        CB    @DCGV,@HG         Is Destination GRAM
        JEQ   TOG               YES, Destination is Gram
        CB    @DCGV,@HV         Is Destination VDP?
        JEQ   TOV               Yes, the Destination is Gram
```

```
TOC     MOV  R1,R5              Dest is CPU Ram, set up R5 with add
        AI   R6,>800            Change R6 opcode to MOVB *R4x,*R5+
        JMP  MOVER3             and go move it!

TOV     LI   R5,VDPWD           Dest is VDP, set up R5 with VDP Write add
        ORI  R1,>4000           set up R0 with VDP address for a Write
        BL   @VDESAD            set up the VDP address
        JMP  MOVER3             and go move it!

TOG     MOV  R12,R5             Dest is Gram, set up R5 with G Write Data add
        AI   R5,>400                .
        BL   @GDESAD            set up the Gram address

* R6 contains one of these:    MOVB *R4,*R5    V to G or G to V or Fill to V or
*                              MOVB *R4+,*R5   C to V or G
*                              MOVB *R4,*R5+   V or G to C or Fill to C
*                              MOVB *R4+,*R5+  C to C

MOVER3  X    R6                 Move the Byte
        DEC  R2                 Decrement # of Bytes to move
        JNE  MOVER3             If not done cotinue moving
MOVER4  BL   @MEMBLK            Else rewrite the memory window
        LWPI WSR                Load WSR
        B    @EDIT3             Return to main program

*-------------------------------VDP to VDP Move
VTOV    BL   @VRDADD            Set VDP Source address
        INC  R0
VTOV1   MOVB *R4,R3             Get the byte
        BL   @VDESAD            Set VDP Destination address
        MOVB R3,*R5             Move the byte
        DEC  R2                 Decrement # of Bytes to move
        JNE  VTOV               If not done continue moving
        JMP  MOVER4             Else return to caller

*-------------------------------Grom/Gram to Gram Move
GTOG    BL   @GSRCAD            Set Grom-Gram Source address
GTOG1   MOVB *R4,R3             Get the byte
        BL   @GDESAD            Set Gram Destination address
        MOVB R3,*R5             Move the byte
        DEC  R2                 Decrement # of Bytes to move
        JNE  GTOG               If not done continue moving
        JMP  MOVER4             Else return to caller

*-------------------------------Grom & VDP address setting subroutines

VDESAD  MOVB @UTIL1,@VDPWA      Set VDP Destination address
        MOVB R1,@VDPWA              .
        INC  R1                 for VDP to VDP move
        RT                      return

GSRCAD  MOVB R0,@>402(R12)      Set Grom/Gram Source address
        MOVB @UTIL0,@>402(R12)      .
        INC  R0                 for Grom/Gram to Gram move
        RT                      return

GDESAD  MOVB R1,@>402(R12)      Set Gram Destination address
        MOVB @UTIL1,@>402(R12)      .
        INC  R1                 for Grom/Gram to Gram move
```

RT

```
*
*****************************************************************************
*
* MEMORY EDITOR FOR THE GRAM KRACKER
*
*       ENTRY: Upper/Lower char. tables loaded in E/A space
*              9901 set for keyboard operation
*              VDP registers set to E/A default
*              GROM library base in R13 of GPL workspace
*              Keyboard area initialized by console keyscan
*
*-----------------------------------------------------------------------------
*
        DEF   START
*
        AORG >2A32
*
START   COPY "WDS1.GKED.START"
        COPY "WDS1.GKED.COLOR"
        COPY "WDS1.GKED.MOVER"
        COPY "WDS1.GKED.MEMDMP"
        COPY "WDS1.GKED.SPCLOPS"
        COPY "WDS1.GKED.DSRLNK"
        COPY "WDS1.GKED.SEARCH"
        COPY "WDS1.GKED.MEDIT"
        COPY "WDS1.GKED.SWPMEM"
        COPY "WDS1.GKED.BIAS"
        COPY "WDS1.GKED.ASCHEX"
        COPY "WDS1.GKED.F-EDITOR"
        COPY "WDS1.GKED.MEMBLK"
        COPY "WDS1.GKED.UTIL"
        COPY "WDS1.GKED.DBASE"
NAMBUF  EQU   $
DNAME   TEXT 'PIO.'
        END
```

```
*
*===============================================================================
*
* ENTRY TO SEARCH ROUTINE
*
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*
SEARCH BL   @CROFF              Turn cursor off
       LWPI UTILWS              Load UTILITY workspace
       MOV  @WSR+30,R15         Grab flags
*
       SZC  @HC,R15             Set key route
       SOC  @H8,R15                .
*
       MOV  @FLDPNR,@TEMP12     Save field pointer for STARTN in PARAM
*
       LI   R0,CGV-6            Disable all fields
       LI   R1,7                   .
SEARC1 SOC  @H8000,*R0             .
       AI   R0,8                   .
       DEC  R1                     .
       JNE  SEARC1                 .
       SZC  @H8000,@AOP-6       Enable AOP and BOP fields
       SZC  @H8000,@BOP-6          .
*
       MOV  @WSR+28,R7          Move R14 into R7
       S    @MADDR,R7           Put cursor address difference into R7
       CB   @HV,@CGV            Is it a VDP search?
       JNE  SEARC2              NO!
       ANDI R7,>3FFF            Mask VDP
HC000  EQU  $+2                 >C000
       LI   R0,>C000            Load mask
       SZC  R0,@AOP             Mask addresses
       SZC  R0,@BOP                .
SEARC2 MOV  R7,@TEMP30          Store difference
*
       MOV  @BLKSCN,@TEMP20     Save BLKSCN
       MOV  @BYTENO,@TEMP21     Save BYTENO
       LI   R1,80               Load R1 with two line lengths
       A    R1,@BLKSCN          Move top of window down two lines
       MOV  @BLKSCN,@TEMP22     Save this value
       LI   R1,-24              Less 24 bytes for two lines
       A    R1,@BYTENO          Subtract 24 bytes from window
       MOV  @BYTENO,@TEMP23     Save this value
*
       MOV  @MADDR,R0           Put MADDR onto screen
       BL   @BIHEXW                .
       LI   R0,TOP+43              .
       LI   R1,HEXW$               .
       LI   R2,4                   .
       BL   @VMBW                  .
*
       LI   R0,TOP+201          Add 3 lines to address
       LI   R1,SLINE            Load single line character
       LI   R2,35               35 chars to make a line
       BL   @RPTBLK             Put line on screen
*
       BL   @MEMBLK             Rewrite memory block
       LI   R1,12               Twelve byte search string
       MOV  R1,@TEMP24          Save 12 in TEMP24
```

```
        MOV   @TEMP20,@BLKSCN   Restore window address
        MOV   @TEMP24,@BYTENO   Set byte count to 12
        MOV   @MADDR,@TEMP26    Save present address and type
        MOV   @CGV,@TEMP27        .
        LI    R7,SRCH$          Move in search string address
        MOV   R7,@MADDR           .
HAC     EQU   $+2
        LI    R7,>6300          Move in type 'c'
        MOV   R7,@CGV             .
        BL    @MEMBLK           Put string up on screen
*
        LI    R1,AOP-6          Load first field for F-EDITOR
        B     @EDIT2            Branch to F-EDITOR
*
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
* Return to status screen
*-----------------------------------------------------------------------------
*
SRCHRT  MOV   @TEMP20,@BLKSCN   Restore previous values for a return
        MOV   @TEMP21,@BYTENO     .
        MOV   @TEMP26,@MADDR      .
        MOV   @TEMP27,@CGV        .
*
        LI    R0,CGV-6          Enable all fields
        LI    R1,7                .
SRCHR4  SZC   @H8000,*R0          .
        AI    R0,8                .
        DEC   R1                  .
        JNE   SRCHR4              .
*
        LI    R7,>1C            Load a reset mask
        SZC   R7,R15            Reset key route and search flag
        MOV   @WSR+30,R7        Fetch previous flag
HC      EQU   $+2
        ANDI  R7,>C             Mask all but key route
        SOC   R7,R15            Set key route in R15
        MOV   R15,@WSR+30       Save flag for STARTN in PARAM
*
        MOV   @TEMP30,R7        Restore cursor in window if
        A     @MADDR,R7          returning there
        CB    @HV,@CGV            .
        JNE   SRCHR1              .
        ANDI  R7,>3FFF            .
SRCHR1  MOV   R7,@WSR+28           .
*
        MOV   @TEMP12,@FLDPNR
        BL    @CROFF            Turn cursor off before changing workspaces
        BL    @MEMBLK           Rewrite memory block
        LWPI  WSR               Load main program workspace
        CZC   @HC,R15           Are we returning to the F-EDITOR?
        JEQ   SRCHR3            YES!
        BLWP  @SPCLL            NO! Go update MADDR
SRCHR3  B     @EDIT3            Turn cursor back on and return
*
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
* Entry and exit between DEFINE STRING and SET ADDRESS
*-----------------------------------------------------------------------------
*
DEFSTR  SOC   @H10,R15          Set SEARCH flag for MEDIT
        MOV   @FLDPNR,@TEMP9    Save FLDPNR
```

```
        BL    @CROFF              Turn cursor off
        MOV   @BLKSCN,R12         Set up MEDIT
        MOV   @MADDR,R14             .
        B     @MEMED3             Go to MEDIT

DEFRTN  SZC   @H10,R15            Reset SEARCH flag
        BL    @CROFF              Turn cursor off
        MOV   @TEMP9,R1           Restore FLDPNR
        B     @EDIT1              Hop into the F-EDITOR
*
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
* Search memory routine
*-------------------------------------------------------------------------------
*
INCR3   DATA  >0583               INC R3
DECR3   DATA  >0603               DEC R3
SRCMEM  BL    @CROFF              Turn cursor off
*
        SZC   @H10,R15            Reset search flag
        MOV   R14,@TEMP25         Save R14
        S     @MADDR,@TEMP25      Fetch cursor/MADDR difference
        INC   @TEMP25             Add one to it
*
        MOV   @TEMP22,@BLKSCN     Restore modified window values
        MOV   @TEMP23,@BYTENO        .
        MOV   @TEMP26,@MADDR         .
        MOV   @TEMP27,@CGV           .
*
        MOV   @AOP,R3              Determine if we're going forward
        MOV   @BOP,R4               or backwards in our search
        MOV   @INCR3,@SRCIN1         .
        MOV   @INCR3,@SRCIN2         .
        C     R3,R4                  .
        JL    SRCME1                 .
        MOV   @DECR3,@SRCIN1         .
        MOV   @DECR3,@SRCIN2         .
*
SRCME1  C     R3,R4               Are we done searching?
        JEQ   SRCME9              YES!
        CB    @HAC,@TEMP27        Is it CPU space
        JNE   SRCM10              NO!
        CI    R3,>8400            Avoid searching memory mapped
        JL    SRCM10               area
        CI    R3,>A000               .
        JL    SRCME6                 .
SRCM10  MOV   R3,R6               Initializations
        LI    R7,SRCH$               .
        MOV   @TEMP25,R13            .
        CB    @HV,@CGV            Is it VDP?
        JNE   SRCME3              NO!
SRCME2  MOV   R6,R0               Move address into R0
        BL    @VSBR               Fetch byte
        CB    R1,*R7+             Is it a match?
        JNE   SRCME6              NO!
        INC   R6                  Increment address
        ANDI  R6,>3FFF            Mask address for wrap around
        DEC   R13                 Decrement string length
        JNE   SRCME2              Loop in not done
        JMP   SRCME7              Got it!
SRCME3  CB    @HG,@CGV            Is it GROM?
```

```
        JNE   SRCME5              NO!
SRCME4  MOV   R6,R0               Same process as with VDP above
        BL    @GWADD                 .
        BL    @GRDAT                 .
        CB    R1,*R7+                .
        JNE   SRCME6                 .
        INC   R6                     .
        DEC   R13                    .
        JNE   SRCME4                 .
        JMP   SRCME7                 .
SRCME5  CB    *R6+,*R7+           Must be CPU search
        JNE   SRCME6                 .
        DEC   R13                    .
        JNE   SRCME5                 .
        JMP   SRCME7                 .
SRCIN1  EQU   $
SRCME6  INC   R3                  Adjust AOP address
        JMP   SRCME1              Loop
*
SRCME7  MOV   R14,@TEMP28          Save R14 of search string
        MOV   R3,@MADDR           Move new address into MADDR
        MOV   R3,R14              Move new address into R14
SRCIN2  EQU   $
        INC   R3                  Adjust address
        MOV   R3,@AOP             Move new address into AOP
        MOV   R3,R0               Move new address into R0 also
        CB    @HV,@CGV            Are we dealing with VDP?
        JNE   SRCM11              NO!
        ANDI  R0,>3FFF            Mask address
        SZC   @HC000,@AOP            .
SRCM11  BL    @BIHEXW             Get HEX of address for screen
        LI    R0,TOP+57           Screen address
        LI    R1,HEXW$            Address of HEX string
        LI    R2,4                Four characters to write
        BL    @VMBW               Put address on screen
*
        BL    @MEMBLK             Rewrite window
        BLWP  @SPCLL
        MOV   @TEMP28,R14          Restore search string cursor address
        MOV   @MADDR,@TEMP26      Store new MADDR in TEMP26
*
SRCME9  LI    R7,SRCH$            Reload search string values
        MOV   R7,@MADDR              .
        MOV   @HAC,@CGV              .
        MOV   @TEMP20,@BLKSCN        .
        MOV   @TEMP24,@BYTENO        .
*
        SOC   @H10,R15            Set search flag for MEDIT
        BL    @CRINIT             Put cursor back up
        B     @KBSCAN             Back to editor
*
*--------------------------------------------------------------------------------
* Subroutine to update MADDR on the screen
*
SPCLL   DATA  DLNKWS,SPCLL1       Entry vectors
SPCLL1  MOV   R14,R0              Save caller's return
        LI    R14,SPCLL2          Load routine address
        RTWP                      Reload caller's workspace
SPCLL2  MOV   @LR3VEC,@TEMP1      Save LR3VEC into TEMP1
        LI    R7,SPCLL3           Load return from MEDIT
```

```
        MOV   R7,@LR3VEC
        B     @LR2                  Go the MEDIT
SPCLL3  MOV   @TEMP1,@LR3VEC        Restore LR3VEC
        MOV   @DLNKWS,R11           Load caller's return
        RT                          Return to caller
```

```
*
*--------------------------------------------------------------------------------
* Update memory block with new memory type
*
CGVCHK JEQ  SPCL2             Skip if invalid entry
       BLWP @SPCL             Set up return to this return from F-EDITOR

CGVC11 CLR  @TEMP1            Clear mask
       CB   @CGV,@HV          Is it VDP we've changed to?
       JNE  CGVCH3            NO!
       MOV  @HC000,@TEMP1     YES! Set VDP mask
CGVCH3 LI   R7,MADDR          Load address of MADDR
       LI   R10,3             Three address to mask. MADDR, AOP and BOP
CGVCH4 SZC  @TEMP1,*R7        Mask value in stack
       MOV  *R7,R0            Grab new value
       BL   @BIHEXW           Convert it to hex
       MOV  R7,R0             Get address of value
       AI   R0,-6             Back up to info/address word
       MOV  *R0,R0            Get info/address word
       ANDI R0,>3FF           Mask of info bits
       LI   R1,HEXW$          Put new value on the screen
       LI   R2,4                 .
       BL   @VMBW                .
H8     EQU  $+2               >8
       AI   R7,8              Increment to next stack entry
       DEC  R10               Have we dealt with all three entries?
       JNE  CGVCH4            NO!
* This section handles the destination value
       CB   @DCGV,@HV         Is destination VDP?
       JNE  CGVCH5            NO!
       SZC  @HC000,@DOP       Mask destination value
       MOV  @DOP,R0           Restore new value to stack
       BL   @BIHEXW           Get hex of new value
       LI   R0,TOP+97         Put new value on screen
       LI   R1,HEXW$             .
       LI   R2,4                 .
       BL   @VMBW                .
CGVCH5 BL   @MEMBLK           Update memory block
       B    @EDIT3            Branch to EDIT3
*
*--------------------------------------------------------------------------------
* Subroutine set up return from F-EDITOR for SPECIAL processing of input
*
SPCL   DATA DLNKWS,SPCLS      Routine vectors
SPCLS  MOV  R14,R0            Save caller's return address in DLNKWS R0
       LI   R14,SPCL1         Load SPCL return address
       RTWP                   Restore workspace and continue processing
SPCL1  MOV  @KRTVEC,@TEMP1    Save KRTVEC
       MOV  @POPVEC,@TEMP28   Save POPRTN vector
       LI   R7,SPCL3          Load SPCL3 vector
       MOV  R7,@KRTVEC           .
       MOV  R7,@POPVEC           .
SPCL2  RT                     Let F-EDITOR finish
SPCL3  MOV  @TEMP1,@KRTVEC    Restore KRTVEC vector
       MOV  @TEMP28,@POPVEC      .    POPVEC
       MOV  @DLNKWS,R11       Restore caller's return vector
       RT                     Return to caller
*
*--------------------------------------------------------------------------------
* SPECIAL OPERATIONS ROUTINE
```

```
* Check field range when looking at a VDP window
*
RANGEA JEQ   SPCL2              Entry for MADDR, AOP and BOP. Skip in invalid
       MOV   @CGV,@TEMP1        Move CGV into TEMP1
       JMP   RANGE1             Go to routine
*
RANGEB JEQ   SPCL2              Entry for DEST. Skip if invalid entry
       MOV   @DCGV,@TEMP1       Move DCGV into TEMP1
*
RANGE1 MOV   R5,R5              Are we on the first character?
       JNE   RANGE2             NO!
       CB    @HV,@TEMP1         Are working with VDP RAM?
       JNE   RANGE2             NO!
       CI    R1,>3400           Is the leading digit less than 3?
       JL    RANGE2             YES!
       LI    R3,HBFF            Indicate an error
       JMP   SPCL2              Don't accept it
*
RANGE2 BLWP  @SPCL              Set up for return to F-EDITOR
       BL    @MEMBLK            Update memory block
       B     @POPSTK            Return to F-EDITOR through POPSTK
```

```
OLDPOS EQU  $
FLDPNR EQU  $+2
VALPNR EQU  $+4
SOPPNR EQU  $+6
DELAY  EQU  $+8
SCRVLU EQU  $+10
FRSTFD EQU  $+12
TIMER  EQU  $+14
TEMP9  EQU  $+16
TEMP12 EQU  $+18
TEMP20 EQU  $+20
TEMP21 EQU  $+22
TEMP22 EQU  $+24
TEMP23 EQU  $+26
TEMP24 EQU  $+28
TEMP25 EQU  $+30
TEMP26 EQU  $+32
TEMP27 EQU  $+34
TEMP28 EQU  $+36
TEMP29 EQU  $+38
TEMP30 EQU  $+40
TEMP31 EQU  $+42
DLNKWS EQU  $+44
OP$    EQU  $+76
*===============================================================================
* STEPPER VDP enviornment restoration
*-------------------------------------------------------------------------------
       BL   @GRADD             Fetch GROM address
       MOV  R0,@SAVGRM         Save it for return
*
       LWPI WSR                Load workspace
       MOV  @H100,R15          Set ASCII flag
       LI   R0,>81F0           Set VDP for text mode
       BL   @VRDADD               .
       LI   R0,>87F4           Set screen color
       BL   @VRDADD               .
*
       LI   R0,DNAME+4         Load address after device name
       LI   R2,144-4           Put 44 spaces into device name area
START1 MOVB @HB20,*R0+         Do it!
       DEC  R2                    .
       JNE  START1                .
*
*-------------------------------------------------------------------------------
* Load TIC,CURSOR,EDGE,SLINE and DLINE characters
*
       LI   R0,>8D8            Load starting position for TIC and CURSOR
       LI   R1,XCHARS          Load CPU address of TIC and CURSOR
       LI   R2,>28             Sixteen bytes to transfer
       BL   @VMBW              Transfer bytes to VDP RAM
*-------------------------------------------------------------------------------
* Clear screen
*
START2 BL   @CLRSCN            Clear screen
*
*-------------------------------------------------------------------------------
* Put up two of the screen dividers
*
SLINE  EQU  >1B00
DLINE  EQU  >1C00
```

```
        LI   R0,TOP+121         Move address up a line
        LI   R1,DLINE           Load double line character
        LI   R2,35              Thirty-five characters to write
        BL   @RPTBLK            Put bottom double memory area line on screen
        LI   R0,TOP+641         Move address up a line
        LI   R1,DLINE           Load double line character
        LI   R2,35              Thirty-five characters to write
        BL   @RPTBLK            Put bottom double memory area line on screen
*
*--------------------------------------------------------------------------------
* Format the screen with fixed text
*
        LI   R1,SCRN            Load SCRN stack address
FORMA1  MOV  *R1+,R0            Load screen location
        JEQ  FORMA2             If zero the return
        MOV  *R1,R3             Move string length into R3
        MOV  *R1+,R2            Load string length
        A    R1,R3              Add address to R3. R3 now points to next entry
        BL   @VMBW              Put string on the screen
        MOV  R3,R1              Restore link
        JMP   FORMA1            Loop until entire stack is written
FORMA2  EQU  $
*
*
ZEROP$  LI   R1,-36             Zero out OP$
        LI   R0,>1D1D              .
ZEROP1  MOV  R0,@OP$+36(R1)        .
        INCT R1                    .
        JNE  ZEROP1                .
        BL   @MEMBLK            Write memory block
*
        B    @EDIT              Branch vector. DYNAMIC!!!
```

```
SWPMMM S     @MADDR,R14        Entry from MEDIT. How far into block is cursor?
       CB    @CGV,@HV          Are we in VDP?
       JNE   SWPME1            NO!
       ANDI  R14,>3FFF         Mask address
       JMP   SWPME1
SWPMEM CLR   R14               Entry from F-EDITOR. Zero out cursor position
SWPME1 BL    @CROFF            Turn cursor off
       MOVB  @CGV,R6           Grab CGV character
       MOV   @MADDR,R7         Grab MADDR
*
       LI    R3,WSTACK         Initialize R3 to top of stack
       LI    R4,WSTAC1         Initialize R4 to middle of stack
*
       MOVB  *R3,@CGV          Rotate stack up and put
       MOV   *R4,*R3+           current values on bottom
       MOV   R6,*R4+               .
       MOV   *R3,@MADDR            .
       A     *R3,R14               .
       MOV   *R4,*R3+              .
       MOV   R7,*R4                .
*
       LI    R0,TOP+42         Load address of FIELD4
       MOVB  @CGV,R1           Move CGV character into R1
       BL    @VSBW             Put new CGV on screen
*
       B     @CGVC11           Return through CGVC11 to update screen
```

```
*
*--------------------------------------------------------------------------------
* KEY OPERATIONS BLOCK
*
* Scan keyboard and branch on a match
*
KEYVE1  INCT  R2                  Increment R2 to next set of two vectors
KEYVEC  MOV   *R2+,R0             Move key value into R0
        JEQ   KEYVE3              If zero then return
        C     R0,R1               Does the key scanned match the value from
*                                  the stack?
        JNE   KEYVE1              NO! Try again
KEYVE2  MOV   *R2,R2              YES! Move branch vector into R2
        B     *R2                 Branch to vector
KEYVE3  RT
*
CHKKEY  MOV   R11,R10             Save return address
        STWP  R1                  Grab present workspace address
        MOV   R1,@KSCVEC          Move it into KSCVEC for recovery later
        LWPI  GPLWS               Load GPL workspace
        BL    @KSCAN              Scan keyboard
KSCVEC  EQU   $+2
        LWPI  WSR                 Load ST workspace
        MOVB  @STATUS,R0          Move STATUS into R0
        CLR   R1                  Clear R1
        MOVB  @KEYBRD,R1          Move KEY into R1
        B     *R10                Return to caller
*
* Scan keyboard column zero. LEAVE: Row # in MSB of R6
*
SCANCT  MOV   R12,R9              Save cursor info in R9
SCANC2  LI    R12,ROWBAS          Load CRU base with column number
        CLR   R6                  Clear R6
        LDCR  R6,3                Load row zero into 9901
        LI    R12,COLBAS          Load CRU base with column number
        SETO  R6                  Clean R6
        STCR  R6,8                Store keyboard row into R6
        MOV   R9,R12              Restore cursor position
        RT                        Return to caller
*
*--------------------------------------------------------------------------------
* VDP OPERATIONS BLOCK
*
VSBW    MOV   R11,R8              Save return address
        BL    @VWDADD             Set VDP address
        MOVB  R1,@VDPWD           Move byte into VDP
        JMP   VDPRT               Return
VSBR    MOV   R11,R8              Save return address
        BL    @VRDADD             Set VDP address
        MOVB  @VDPRD,R1           Read byte from VDP
        JMP   VDPRT               Return
VMBW    MOV   R11,R8              Save return address
        BL    @VWDADD             Set VDP address
VMBW1   MOVB  *R1+,@VDPWD         Move byte into VDP
        DEC   R2                  Decrement loop counter
        JNE   VMBW1               Jump if not finished
        JMP   VDPRT               Return
VMBR    MOV   R11,R8              Save return address
        BL    @VRDADD             Set VDP address
VMBR1   MOVB  @VDPRD,*R1+         Read data from VDP
```

```
        DEC   R2                  Decrement loop counter
        JNE   VMBR1               Loop if not finished
VDPRT   B     *R8                 Return
VWDADD  ORI   R0,>4000            Set write data bit
VRDADD  SWPB  R0                  Swap LSB into place
        MOVB  R0,@VDPWA           Write LSB of address
        SWPB  R0                  Swap MSB into place
        MOVB  R0,@VDPWA           Write MSB of address
        ANDI  R0,>3FFF
        RT                        Return
*
*--------------------------------------------------------------------------------
* Cursor routine
*
CRINIT  MOV   R11,R10             Entry point for CURSOR initization. Save return
        JMP   CRINI1
*
CROFF   MOV   R11,R10             Entry point for CURSOR OFF routine. Save return
        SZC   @H1,R15             Reset cursor flag
        JMP   CROFF1              Take cursor off the screen
*
CURSOR  MOV   R11,R10             Save return address
        C     R12,@OLDPOS         Have we changed position since the last access?
        JNE   CURSO1              YES!
        DEC   @TIMER              NO! decrement timer. Is it >FFFF yet?
        JGT   CURSO3              NO! Return to caller
        XOR   @H1,R15             Toggle cursor flag
        LI    R1,CRSOR            Assume we need to put the cursor up
        CZC   @H1,R15             Was the cursor already up?
        JNE   CURSO2              NO! Put cursor up and reset timer
CROFF1  MOVB  @SCRVLU,R1          Load character into R1
        JMP   CURSO2              Put character on screen and reset timer
*
CURSO1  MOV   @OLDPOS,R0          Load previous screen position into R0
        MOVB  @SCRVLU,R1          Load previous character into R1
        BL    @VSBW               Put previous character back into place on screen
CRINI1  MOV   R12,R0              Move new position into R0
        BL    @VSBR               Get character at new position
        MOVB  R1,@SCRVLU          Move new character into SCREEN VALUE
        MOV   R12,@OLDPOS         Move new position into OLD POSITION variable
        CB    R1,@HEDGE           Are we sitting on an EDGE character?
        JNE   CURSO4              NO!
        A     R2,R12              YES! Add/Subtract one from R12
        JMP   CRINI1              Move cursor over one
CURSO4  SOC   @H1,R15             Set cursor flag
        LI    R1,CRSOR            Load cursor into R1
CURSO2  MOV   R12,R0              Move current screen address into R0
        BL    @VSBW               Put cursor on new screen position
        LI    R1,TCOUNT           Load TIMER count into R1
        MOV   R1,@TIMER           Restore TIMER
CURSO3  B     *R10                Return to caller
*
*--------------------------------------------------------------------------------
* Clear screen and repeat block routines
*
RPTBLK  MOV   R11,R10             Save return address
        JMP   CLRSC1
CLRSCN  MOV   R11,R10             Save return address
        CLR   R0                  Load screen address
        LI    R1,>1D00            Load an EDGE character into R1
```

```
       LI   R2,959             Load R2 with number of spaces left to clear
CLRSC1 BL   @VSBW              Write first space to set VDP address
CLRSC2 MOVB R1,@VDPWD          Move spaces onto the screen
       DEC  R2                 Are we done?
       JNE  CLRSC2             NO!
       B    *R10               Return to caller
*
*==============================================================================
*      GROM OPERATIONS BLOCK
*
GWADD  MOV  @>83FA,R2     Set up for GROM address
       MOVB R0,@>402(R2)  Set GROM address
       SWPB R0
       MOVB R0,@>402(R2)
       SWPB R0
       RT                 Return to caller
*
GRADD  MOV  @>83FA,R2     Set up for GROM address
       MOVB @2(R2),R0     Set GROM address
       SWPB R0
       MOVB @2(R2),R0
       SWPB R0
       DEC  R0
       RT                 Return to caller
*
GRDAT  MOV  @>83FA,R2     Set up for GROM address
       MOVB *R2,R1        Read GROM byte
       RT                 Return to caller
*
GWDAT  MOV  @>83FA,R2     Set up for GROM address
       MOVB R1,@>400(R2)  Write GROM byte
       RT                 Return to caller
*
*==============================================================================
* KEYBOARD SCAN
*
ROWBAS EQU  >24
COLBAS EQU  >06
OLDMOD EQU  >83C7
DBNCE  EQU  >83C8
*
* Perform some initializations
*
KSCAN  LI   R1,5               Set row counter
       CLR  R2                 Assume no key down at present
       CLR  R6                    .
       CLR  R7                 Assume no modifiers (CTRL, FNTN and SHIFT)
       CLR  R12
       SBO  21                 Turn alpha lock line off
*
* Scan the keyboard for a key
*
ROWLP  LI   R12,ROWBAS         CRU base for row selection
       SWPB R1                 Move row number into place
       LDCR R1,3               Turn specified row on
       SWPB R1                 Restore row counter
       LI   R12,COLBAS         CRU base for column selection
       SETO R4                 Clean column register
       STCR R4,8               Fetch column data
       INV  R4                 Make one's represent keys down
```

```
        MOV   R1,R1             Are we on row 0?
        JNE   NOTONE            NO!
        MOVB  R4,R7             Save CTRL and/or FCTN keys
        ANDI  R4,>0F00          Mask out CTRL and/or FCTN bits

NOTONE  MOVB  R4,R4             Do we have a key?
        JEQ   NXTROW            NO! Try next line
*
* Determine which key is depressed
*
GOTKEY  MOV   R2,R2             Is this the second key being held down?
        JNE   NXTROW            YES! Just ignore it
        SETO  R2                Indicate that the first key press is found
        MOV   R1,R3             Get the row number
        SLA   R3,3              Multiply it by 8
        DEC   R3                Adjust it for the INC that follows
CNTLP   INC   R3                Add in a column
        SLA   R4,1              Shift the key bit into the carry
        JNC   CNTLP             Loop if we have'nt hit the key bit yet
        MOV   R1,R1             Are we working with line 0?
        JEQ   NXTROW            YES!
        LI    R1,1              NO! Force next line to be zero
*
* Select next line to be polled
*
NXTROW  DEC   R1                Decrement row number
        JOC   ROWLP             If not finished then go scan next line
        MOV   R2,R2             Was a key pressed this time through?
        JNE   DEBOUN            YES!
*
* No keys are down so wrap things up
*
NOKEY   CLR   R6                No key so reset flag register
        MOVB  R6,@OLDMOD        Clear any old modifiers
H700    EQU   $
        SETO  R0                Load R0 with >FFFF
        CB    R0,@DBNCE         Was a key just released?
        JEQ   NOKEY2            NO!
        LI    R12,1250          YES! Do some debounce. Load 10mS
KIL1    DEC   R12               Loop and kill some time
        JNE   KIL1                 .

NOKEY2  MOVB  R0,@DBNCE         Clear keyboard debounce register
        JMP   OLDCHR            Go indicate keyboard state to user
*
* A key is down so do a debounce if necessary
*
DEBOUN  CB    @R3LB,@DBNCE      Is the same key station depressed as last time?
        JEQ   MODIFY            YES!
H2000   EQU   $+2
HB20    EQU   H2000
        LI    R6,>2000          Load new key flag
        LI    R12,1250          Load 10mS of debounce time
KIL2    DEC   R12               Loop for awhile
        JNE   KIL2                 .
        MOVB  @R3LB,@DBNCE      Indicate which keyboard station is depressed

NEWMOD  MOVB  R7,@OLDMOD        Move modifiers into modifier register
*
* We have the key station now get the key code from tables
```

```
*
MODIFY MOVB @OLDMOD,R7         Fetch modifiers
       LI   R1,KFNCTN          Load FCTN character table address
       SLA  R7,2               Is the FCTN key down?
       JOC  MAPIT              YES!
       LI   R1,KFNCTN          Load FCTN character table address
       SRL  R7,15              Is the FCTN key down?
       JOC  MAPIT              YES!
       LI   R1,KSHIFT          Load the SHIFT character table address
       DEC  R7                 Is the SHIFT key(s) down?
       JEQ  MAPIT              YES!
       LI   R1,KEYTAB          Must be unmodified keyboard!

MAPIT  A    R3,R1              Copy table offset into R1
       MOVB *R1,R0             Get key value
*
* Here's where we check for the alpha lock
*
       LI   R12,'az'           Load lower character set range
       CB   R0,R12             Is key value less than an 'a'?
       JL   RSTP5              YES!
       CB   R0,@R12LB          Is key value greater than a 'z'?
       JH   RSTP5              YES!
       CLR  R12                Reset CRU base
       SBZ  21                 Turn alpha lock line on
       SRC  R12,14             Waste some time
       TB   7                  Is alpha lock down?
       JEQ  RSTP5              NO!
       SB   @H2000,R0          Map lower case into upper case
RSTP5  SBO  21                 Turn alpha lock line off
*
* Move key value out and set/reset STATUS

OLDCHR MOVB R0,@KEYBRD         Move key value (or >FF) into key register
       MOVB R6,@STATUS         Load new key flag (if any) into GPL STATUS byte
       RT                      Return to caller
*
*===============================================================================
*
* Keyboard tables
*
KEYTAB DATA >FFFF,>FFFF,>FF0D,>203D '***** ='
       DATA >7877,>7332,>396F,>6C2E 'xws29ol.'
       DATA >6365,>6433,>3869,>6B2C 'ced38ik,'
       DATA >7672,>6634,>3775,>6A6D 'vrf47ujm'
       DATA >6274,>6735,>3679,>686E 'btg56yhn'
       DATA >7A71,>6131,>3070,>3B2F 'zqa10p;/'
KSHIFT DATA >FFFF,>FFFF,>FF0D,>202B '****** +'
       DATA >5857,>5340,>284F,>4C3E 'XWS@(OL>'
       DATA >4345,>4423,>2A49,>4B3C 'CED##IK<'
       DATA >5652,>4624,>2655,>4A4D 'VRF$&UJM'
       DATA >4254,>4725,>5E59,>484E 'BTG%^YHN'
       DATA >5A51,>4121,>2950,>3A2D 'ZQA!)P:-'
KFNCTN DATA >FFFF,>FFFF,>FF0D,>2005 '****** *'
       DATA >0A7E,>0804,>0F27,>C2B9 '*~***'**'
       DATA >600B,>0907,>063F,>C1B8 ''****?**'
       DATA >7F5B,>7B02,>015F,>C0C3 '*[{**_**'
       DATA >BE5D,>7D0E,>0CC6,>BFC4 '*]}*****'
       DATA >5CC5,>7C03,>BC22,>BDBA '\*|**"**'
*
```

```
*****************************************************************************
*
* BINARY TO HEXIDECIMAL CONVERSION FOR X-BASIC
*
*      ENTER: R0 contains the hex value to be converted to displayable chars.
*             First entry point is BIHEXB for byte values.  The value to be
*              converted should be in the MSB.
*             Second entry point is BIHEXW for word values.
*
*      LEAVE: Characters are in HEXB$ for byte values and in HEXW$ for word
*              values.  Original valued is SWPBed in R0 in the case of a
*              byte operation and is restored in a word operation
*
*      USES : R0,R1,R2,R11
*
*---------------------------------------------------------------------------
*
HEXB$  EQU  $+2
HEXW$  DATA 0,0
STR$   DATA >3031,>3233,>3435,>3637,>3839,>4142,>4344,>4546
BIHEXB LI   R1,-2
       JMP  BIHEX

BIHEXW LI   R1,-4

BIHEX  SRC  R0,12
       MOV  R0,R2
       ANDI R2,>F
       MOVB @STR$(R2),@HEXW$+4(R1)
       INC  R1
       JNE  BIHEX
       RT
```