```
*** FUNCTIONALITY TO ADD:
***
*** -- IDE ERROR command - show error byte

*OVERVIEW
*--------

*IDE command scanner run from TIBUG by TIBUG doing B @STRT. Return to TIBUG by doing B @>0080.
*IDE loads its own workspace on entry, and TIBUG reloads its workspace on return.

*BASIC needs to run individual IDE routines, in their own workspace. So call the individual
*routine, both from the command scanner and BASIC, by doing a BLWP. From the command scanner,
*the routine will use the same workspace as already being used by the command scanner.



*IDE IDENTIFY
*Read drive info into 1st sector buffer and display.
*Read sector 0 into 1st sector buffer and display drive name, number of sectors and number sectors free.

*IDE STATUS
*Read status information from drive and display.

*IDE SPIN DOWN/UP
*Power drive down/up.

*IDE SHOW SECTOR
*Input sector number to read.
*Read sector into 1st sector buffer and display.

*IDE SECTOR FILL
*Input start sector number.
*Input number of sectors.
*Input fill value.
*Fill 1st sector buffer and write to disk for each sector to fill.

*IDE SAVE
*Input file name.
*Read sector 1 into 1st sector buffer.
*Step through sector 1 and read each file descriptor record into 2nd sector buffer and check if
*  file name already exists. Prompt whether to overwrite file if so. If not to overwrite file,
*  return and prompt for file name. If to overwrite file, save pointer to original file's FDR.
*Input start, end, execute addresses and build new FDR in 2nd sector buffer.
*Calculate number of bytes to save and number of sectors required.
*Read sector 0 into 1st sector buffer and check disk space available.
*  Display error message abort if not.
*Input program description and add to new FDR in 2nd sector buffer.
*Find 1st free sector in sector bitmap in sector 0 cached in 1st sector buffer and use this for
*  storing the FDR.
*Find next free sector in sector bitmap in sector 0 cached in 1st sector buffer.
*  Copy data to 3rd sector buffer and write to sector.
*  Repeat until written all data.
*Write sector 0 cached in 1st sector buffer back to disk with updated sector bitmap.
*Write FDR in 2nd sector buffer to sector reserved earlier.
*Read sector 1 into 1st sector buffer, add address of FDR and write back to disk.
*If overwriting file, retrieve pointer to original file's FDR, read the FDR into the 2nd sector
*  buffer and and jump to IDE DELETE routine.

*IDE DIR
*Read sector 1 into 1st sector buffer.
*Step through sector 1 and read each file descriptor record into 2nd sector buffer.
*  Display file name plus other details for each file.

*IDE LOAD
*Input file name.
*Check if file name exists, exit if not. This:
*  Reads sector 1 into 1st sector buffer.
*  Reads FDRs into 2nd sector buffer.
*Read sectors into 1st sector buffer and transfer to memory as required.

*IDE RUN
*IDE RENAME
*IDE PROGRAM DESCRIPTION

*IDE DELETE
*Input file name.
*Read sector 1 into 1st sector buffer.
*Step through sector 1 and read each file descriptor record into 2nd sector buffer and check if
*  file name exists. Display error message and abort if not.
*Clear the pointer to the FDR in sector 1, and write sector 1 (in 1st sector buffer) back to disk.
*Read sector 0 into 1st sector buffer.
*Mark FDR sector plus all data sectors used by the file as free in the sector 0 bitmap.
*Write sector 0 bitmap back to disk.

*IDE FORMAT DISK
*IDE RENAME DISK
```

```
*------------------------------------------------------------------------------
*Disk logical organisation
*------------------------------------------------------------------------------
*Sector 0 - Volume Information Block (VIB)
*------------------------------------------------------------------------------
*Bytes >000 - >013: 20 character disk name. Name length defined by the
*                   equate MAXDNLN. Changing MAXDNLN will change the offsets
*                   of the following parameters.
*Bytes >014 - >015: Number of free sectors on the disk. This will be >F50
*                   when the disk is initially formatted.
*Bytes >016 - >1FF: Sector bitmap. Each bit indicates whether the associated
*                   sector is used or not (starting at sector 2). The MS bit
*                   of each word indicates the lower sector number. So if
*                   byte >014 has the value >C0 then sectors 2 and 3 are in
*                   use. If byte >015 has the value >03 then sectors 16 and
*                   17 are in use.
*------------------------------------------------------------------------------
*Sector 1 - List of Pointers to File Descriptor Records (FDRs)
*------------------------------------------------------------------------------
*
*                   List of pointers (sector numbers) to the FDRs. Each
*                   pointer is 2 bytes. Sector 1 is initialised to all
*                   zeroes when the disk is formatted. A non-zero data word
*                   is therefore a pointer to an FDR. Zero values may be
*                   intermingled with the pointers if files are deleted;
*                   these may be used for pointers to new FDRs if new files
*                   are saved.
*
*------------------------------------------------------------------------------
*Sector 2 onwards - File Descriptor Records (FDRs)
*------------------------------------------------------------------------------
*Bytes >000 - >013: 20 character file name. Name length defined by the
*                   equate MAXFNLN. Changing MAXFNLN will change the offsets
*                   of the following parameters.
*Bytes >014 - >015: Program load address.
*Bytes >016 - >017: Program execute address.
*Bytes >018 - >019: Program end address. This is only used by the DIR
*                   command to show the end address to the user.
*Bytes >01A - >01B: Size of file in bytes.
*Bytes >01C - >061: 70 character program description. Description length
*                   defined by the equate MAXFDLN.

*Bytes >062 - >1FF: List of sectors in which the file is stored. The maximum
*                   size of file for the TM 990 is 65536 bytes, which would
*                   require 128 512-byte sectors to store. WHAT ABOUT DATA
*                   FILES THAT COULD EXCEED 64K? The code relies on this
*                   list being zero-terminated.

*Room for time/date stamp?
* - one byte for day (1-31)
* - one byte for month (1-12)
* - one byte for year (00-99)
* - one byte for hours (0-23)
* - one byte for minutes (0-59)
* - one byte for seconds (0-59)

*------------------------------------------------------------------------------




        AORG >1000

*------------------------------------------------------------------------------
*Start here by clearing screen, printing welcome message and prompt.
*------------------------------------------------------------------------------

STRT    LWPI WSREG

        CLR @CBASSA      Clear Cortex BASIC load/save start address such that we can tell a routine
*                        has been called from the IDE command scanner.

        XOP @CLSCRN,14   Clear screen.
        XOP @WELCME,14   Print welcome message.
        XOP @PNTCR,14    Print <CR><LF>.
STRT01  XOP @PNTCR,14    Print <CR><LF>.
STRT02  XOP @PNTPMT,14   Print prompt.

*------------------------------------------------------------------------------
*Accept command at prompt.
```

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

        LI R1,CLMAX      Maximum length of command line allowed.
        LI R3,' '*256    Use <Space> character to display an invisible text entry field.
        BL @TSENTRY      Accept text string.

*--------------------------------------------------------------------------
*Find command in command list and process.
*--------------------------------------------------------------------------

        JEQ STRT02       Text entry length was zero - print command prompt and wait for next command.

FDCD01  LI R2,CLSTART    Pointer to first command structure.
FDCD04  MOV R2,R3
        INCT R3          R3 points to BLWP vector for this command.
        MOV R3,R4
        INCT R4          R4 points to name length for this command.
        MOV R4,R5
        INCT R5          R5 points to name text for this command.

        C *R4,R1         Does this command have the same length as the command entered?
        JNE FDCD02       No, go to next command structure.

        MOV R1,R6        Copy command line entry length.
        DEC R5           R5 now points to the byte before the command name text.
        A R6,R5          R5 now points to the last character in the command name text.

FDCD05  CB *R5,@CLBUFF-1(R6) Does the n'th character in name and command entered match?
        JNE FDCD02       No, go to next command structure.

        DEC R5           Check next (previous) character in the name.
        DEC R6
        JNE FDCD05

        MOV *R3,R3       Names match, execute the code.
        BLWP *R3
        JMP STRT01       Routine returns here. Print command prompt and wait for next command.

FDCD02  MOV *R2,*R2      Reached last command structure in list?
        JEQ FDCD03       Yes, command entered was not found.

        MOV *R2,R2       Get address of next command structure in list.
        JMP FDCD04       Process next command structure.

FDCD03  XOP @PNTERR1,14  Print error message.
        JMP STRT01       Print command prompt and wait for next command.

*--------------------------------------------------------------------------
*Command list.
*--------------------------------------------------------------------------

CLSTART DATA CLHELP      Pointer to next command structure, or >0000 if last command.
        DATA CLS         Pointer to start of code for this command.
        DATA 3           Command name length.
        TEXT 'CLS'       Command name text.
        EVEN

CLHELP  DATA CLIDEID
        DATA HELP
        DATA 1
        TEXT '?'
        EVEN

CLIDEID DATA CLIDEST
        DATA IDEID
        DATA 8
        TEXT 'IDENTIFY'
        EVEN

CLIDEST DATA CLIDESD
        DATA IDEST
        DATA 6
        TEXT 'STATUS'
        EVEN

CLIDESD DATA CLIDESU
        DATA IDESD
        DATA 9
        TEXT 'SPIN DOWN'
        EVEN

CLIDESU DATA CLIDESS
        DATA IDESU
        DATA 7
        TEXT 'SPIN UP'
        EVEN
```

```
CLIDESS DATA CLIDESF
        DATA IDESS
        DATA 11
        TEXT 'SHOW SECTOR'
        EVEN

CLIDESF DATA CLIDESV
        DATA IDESF
        DATA 11
        TEXT 'SECTOR FILL'
        EVEN

CLIDESV DATA CLIDEDR
        DATA IDESV
        DATA 4
        TEXT 'SAVE'
        EVEN

CLIDEDR DATA CLIDELD
        DATA IDEDR
        DATA 3
        TEXT 'DIR'
        EVEN

CLIDELD DATA CLIDERN
        DATA IDELD
        DATA 4
        TEXT 'LOAD'
        EVEN

CLIDERN DATA CLIDRNF
        DATA IDERN
        DATA 3
        TEXT 'RUN'
        EVEN

CLIDRNF DATA CLIDRPD
        DATA IDERNF
        DATA 6
        TEXT 'RENAME'
        EVEN

CLIDRPD DATA CLIDEDL
        DATA IDERPD
        DATA 11
        TEXT 'DESCRIPTION'
        EVEN

CLIDEDL DATA CLIDEFM
        DATA IDEDL
        DATA 6
        TEXT 'DELETE'
        EVEN

CLIDEFM DATA CLIDERM
        DATA IDEFM
        DATA 11
        TEXT 'FORMAT DISK'
        EVEN

CLIDERM DATA CLQUIT
        DATA IDERM
        DATA 11
        TEXT 'RENAME DISK'
        EVEN

CLQUIT  DATA >0000       End of linked list.
        DATA IDEQT
        DATA 4
        TEXT 'QUIT'
        EVEN

*--------------------------------------------------------------------------
*QUIT command.
*--------------------------------------------------------------------------

IDEQT   DATA WSREG
        DATA >0080       Entry point for TIBUG. TIBUG immediately loads its own workspace.

*--------------------------------------------------------------------------
*CLS command.
*--------------------------------------------------------------------------
*Simply clears the screen.
*--------------------------------------------------------------------------

CLS     DATA WSREG       BLWP vector for this command.
        DATA $+2
```

```
        XOP @CLSCRN,14   Clear screen.

        RTWP             Return.

*--------------------------------------------------------------------------
*? (Help) command.
*--------------------------------------------------------------------------
*Steps through the linked list of commands and prints the name of
*each command.
*--------------------------------------------------------------------------

HELP    DATA WSREG       BLWP vector for this command.
        DATA $+2

        XOP @PNTCSP1,14  Print intro text.

        LI R2,CLSTART    Pointer to first command structure.
HELP04  MOV R2,R4
        C *R4+,*R4+      Increment R4 by 4 to point to name length for this command.
        MOV R4,R5
        INCT R5          R5 points to name text for this command.

        MOV *R4,R4       Get the actual name length.

HELP01  MOVB *R5+,R1     Get character from name.
        XOP R1,12        Print character from name.
        DEC R4
        JNE HELP01       Loop for all characters in name.

        XOP @PNTCSP2,14  Print <CR><LF> plus spaces before next name.

        MOV *R2,*R2      Reached last command structure in list?
        JEQ HELP03

        MOV *R2,R2       Get address of next command structure in list.
        JMP HELP04       Process next command structure.

HELP03  RTWP             Return.

*--------------------------------------------------------------------------
*IDE IDENTIFY command.
*--------------------------------------------------------------------------
*Prints the drive name (stored in the drive firmware) and total number of
*sectors on the drive. Then prints the (formatted) disk name, number of
*sectors supported and number of sectors free.
*--------------------------------------------------------------------------

IDEID   DATA WSREG       BLWP vector for this command.
        DATA $+2

*Read drive information from drive and display the drive name and total number
*of sectors on the drive.

        MOV @IDECMID,@IDEPM03  Set up command "Drive ID".
        MOV @IDEBFA1,@IDEPM04  Read into first sector buffer.

        BL @IRWBUFF      Read from drive into sector buffer.

        XOP @PNTDID1,14  Print drive name intro text.

        LI R1,IDEBUF1+>36  Drive name starts >36 bytes into sector buffer.
IDEID01 SWPB *R1         Print two characters (bytes) from word, LS byte first.
        XOP *R1,12
        SWPB *R1
        XOP *R1,12
        INCT R1          Address next word in buffer.
        CI R1,IDEBUF1+>36+40  Done all 40 characters of drive name?
        JNE IDEID01

        XOP @PNTDID2,14  Print drive number sectors intro text.

        LI R1,IDEBUF1+>74  Drive number of sectors starts >72 bytes into the sector buffer.
        SWPB *R1         Print word >74 first, then word >72, second byte of each word first.
        XOP *R1,10
        DECT R1
        SWPB *R1
        XOP *R1,10

*Read sector 0 from the drive and display the disk name, number of sectors
*supported and number of sectors free.

        BL @RDSEC0       Read sector 0 into first sector buffer.

        XOP @PNTDID3,14  Print disk name intro text.

        LI R1,IDEBUF1    Disk name starts at beginning of sector buffer.
```

```
IDEID02 XOP *R1,12       Print two characters (bytes) from word, MS byte first.
        SWPB *R1
        XOP *R1,12
        INCT R1          Address next word in buffer.
        CI R1,IDEBUF1+MAXDNLN  Done all characters of disk name?
        JNE IDEID02

        XOP @PNTDID4,14  Print number of sectors supported intro text.

        LI R1,512-MAXDNLN-2*8  Total number of sectors supported (see IDE FORMAT
*                              DISK command comments for this calculation).
        XOP R1,10        Print number of sectors supported.

        XOP @PNTDID5,14  Print number of sectors free intro text.

        XOP @IDEBUF1+MAXDNLN,10 Print number of sectors free from sector buffer.

BSTRT3  XOP @PNTCR,14    Print <CR><LF>.
        RTWP             Return.

*--------------------------------------------------------------------------
*IDE STATUS command.
*--------------------------------------------------------------------------
*Prints the current state of each bit of the drive status register.
*--------------------------------------------------------------------------

IDEST   DATA WSREG       BLWP vector for this command.
        DATA $+2

        MOV @IDEMD,@IDEWR6  Select master drive.

        MOV @IDERR7,R1   Read status from IDE register 7.

        LI R2,'0'*256    ASCII '0'.
        LI R3,'1'*256    ASCII '1'.

        CLR R4           Counter through 8 status bits.

IDEST03 MOV @PNTIST(R4),R5  Get address of text for this bit from address table.
        XOP *R5,14       Print text for status bit.
        SLA R1,1         Shift IDE status bit into the processor Carry status bit.
        JOC IDEST01      Jump if IDE status bit was set (1).
        XOP R2,12        Print ASCII '0'.
        JMP IDEST02
IDEST01 XOP R3,12        Print ASCII '1'.

IDEST02 INCT R4          Loop for next bit. Increment by 2 because R4 used as offset into
*                        text address table.

        CI R4,8*2        Done all 8 bits?
        JNE IDEST03      No, loop.

        XOP @PNTCR,14    Print <CR><LF>.
        RTWP             Return.

*--------------------------------------------------------------------------
*IDE SPIN DOWN command.
*--------------------------------------------------------------------------
*Simply spins down the disk.
*--------------------------------------------------------------------------

IDESD   DATA WSREG       BLWP vector for this command.
        DATA $+2

        MOV @IDEMD,@IDEWR6  Select master drive.

        BL @IWTRDY       Wait until drive is ready and not busy.

        MOV @IDEMD,@IDEWR7  Load command "Power down drive".

        RTWP             Return.

*--------------------------------------------------------------------------
*IDE SPIN UP command.
*--------------------------------------------------------------------------
*Simply spins up the disk.
*--------------------------------------------------------------------------

IDESU   DATA WSREG       BLWP vector for this command.
        DATA $+2

        MOV @IDEMD,@IDEWR6  Select master drive.

        BL @IWTRDY       Wait until drive is ready and not busy.

        MOV @IDECMPU,@IDEWR7  Load command "Power up drive".
```

```
        RTWP            Return.

*--------------------------------------------------------------------------
*IDE SHOW SECTOR command.
*--------------------------------------------------------------------------
*Requests a sector number from the user, and prints the contents
*of that sector in hex and ASCII.
*--------------------------------------------------------------------------

IDESS   DATA WSREG      BLWP vector for this command.
        DATA $+2

        XOP @PNTGSN1,14 Print prompt for sector number.

        XOP R2,9        Read hexadecimal word from terminal into R2.
        DATA IDESS12    Exit if null entered.
        DATA IDESS12    Exit if non-hex entered.

IDESS09 XOP @PNTCR,14   Print <CR><LF>.

        MOV @H0100,@IDEPM01  Set up number of sectors to read in MS byte.
        MOV R2,@IDEPM02  Set up sector number to read.
        MOV @IDECMRD,@IDEPM03  Set up command "read sectors with retry".
        MOV @IDEBFA1,@IDEPM04  Read into first sector buffer.

        BL @IRWBUFF     Read from drive into sector buffer.

        XOP @PNTGSN2,14 Print text and sector number followed by colon.
        XOP R2,10
        XOP @PNTGSN1+26,12

        XOP @PNTCR2,14  Print 2x<CR><LF>.

        XOP @PNTGSN3,14 Print sector data header.

        LI R7,512/16    Do 32 lines of 16 bytes per line = 512 bytes = 1 sector.

        CLR R1          Used to display address offset within sector. Also used as index
*                       into sector buffer.

IDESS06 LI R5,CLBUFF    Set position of buffer in which to build up ASCII string.
*                       Reuse command line buffer.

        XOP @PNTPMT,14  Print '>' character ...
        XOP R1,10       ... followed by the address ...
        XOP @PNTDID1+29,14  ... followed by two spaces.

        LI R4,8         Set number of words displayed per line.

IDESS05 XOP @IDEBUF1(R1),10  Print word from sector buffer ...
        XOP @PNTDID1+29,12  ... followed by a space.

        LI R8,2         Need to do the following block twice - once for the MS byte, again
*                       for the LS byte.

        MOV @IDEBUF1(R1),R6  Copy word and check if MS byte is within an ASCII displayable range.
IDESS07 ANDI R6,>FF00   Isolate MS byte.
        CI R6,32*256    ASCII value less than 32?
        JL IDESS01      Yes, jump.
        CI R6,126*256   ASCII value greater than 126?
        JH IDESS01      Yes, jump.
        JMP IDESS02     Jump, ASCII value in displayable range.
IDESS01 LI R6,'.'*256   Set to out-of-range character ('.').
IDESS02 MOVB R6,*R5+    Move ASCII character to buffer.

        MOV @IDEBUF1(R1),R6  Copy word again to check if LS byte is within an ASCII displayable range.
        SWPB R6         Move LS byte into MS byte position.
        DEC R8          Done both bytes?
        JNE IDESS07     No, loop.

        INCT R1         Increment index to next word in sector buffer.

        DEC R4          Decrement number of words processed on this line.
        JNE IDESS05     Done a whole line? No, jump to do next word.

        XOP @PNTDID1+29,12  Print one space.

        CLR R6          Write >00 to end of ASCII buffer.
        MOVB R6,*R5+

        XOP @CLBUFF,14  Print ASCII text string built up in buffer.
        XOP @PNTCR,14   Print <CR><LF>.

        DEC R7          Done the required number of lines?
        JNE IDESS06     No, loop to display next line.

        XOP @PNTGSN4,14 Print prompt to press a key for previous sector, next sector, or quit.
```

```
IDESS11 XOP R7,13        Wait for user to press a key.

        CI R7,'P'*256    'P' (Previous) key pressed?
        JNE IDESS08      No, jump to check next key.
        DEC R2           Yes, decrement sector number.
        JMP IDESS09      Jump to display new sector.

IDESS08 CI R7,'N'*256    'N' (Next) key pressed?
        JNE IDESS10      No, jump to check next key.
        INC R2           Yes, increment sector number.
        JMP IDESS09      Jump to display new sector.

IDESS10 CI R7,'Q'*256    'Q' (Quit) key pressed?
        JNE IDESS11      No, jump to wait for another key press.

IDESS12 XOP @PNTCR,14    Print <CR><LF>.
        RTWP             Return.

*--------------------------------------------------------------------------
*IDE SECTOR FILL command.
*--------------------------------------------------------------------------
*Requests a start sector number and number of sectors from the user,
*plus a hex word value, and fills the specified sectors with the
*specified value.
*--------------------------------------------------------------------------

IDESF   DATA WSREG       BLWP vector for this command.
        DATA $+2

        XOP @PNTGSF1,14  Print prompt for start sector number.

        XOP R2,9         Read hexadecimal word from terminal into R2.
        DATA IDESF04     Exit if null entered.
        DATA IDESF04     Exit if non-hex entered.

IDESF01 XOP @PNTGSF2,14  Print prompt for number of sectors.

        XOP R3,9         Read hexadecimal word from terminal into R3.
        DATA IDESF04     Exit if null entered.
        DATA IDESF04     Exit if non-hex entered.

        XOP @PNTGSF3,14  Print prompt for fill value.

        XOP R4,9         Read hexadecimal word from terminal into R4.
        DATA IDESF04     Exit if null entered.
        DATA IDESF04     Exit if non-hex entered.

        LI R1,IDEBUF1    Address of sector buffer to use.
IDESF02 MOV R4,*R1+      Fill sector buffer with specified value.
        CI R1,IDEBUF1+512
        JNE IDESF02

        MOV @H0100,@IDEPM01  Set up number of sectors to write at a time in MS byte.
IDESF03 MOV R2,@IDEPM02  Set up sector number to fill.
        MOV @IDECMWR,@IDEPM03  Set up command "write sectors with retry".
        MOV @IDEBFA1,@IDEPM04  Write from first sector buffer.

        BL @IRWBUFF      Write from sector buffer to drive.

        INC R2           Increment current sector number.
        DEC R3           Decrement number of sectors to fill.
        JNE IDESF03

IDESF04 XOP @PNTCR,14    Print <CR><LF>.
        RTWP             Return.

*--------------------------------------------------------------------------
*IDE SAVE command.
*--------------------------------------------------------------------------
*Requests the following from the user:
*-- File name.
*-- Memory address to start saving from.
*-- Memory address to end saving at (addresses inclusive).
*-- Program execute address. Defaults to start address if not specified.
*-- Optional program description.
*Before saving, checks that the specified file name does not already exist,
*and that there is sufficient room on the disk to save the file. Then saves
*the file. If the specified file does already exist, prompts whether to
*overwrite the file, and if so saves the new file then deletes the original
*file.

*When called from Cortex BASIC, BASIC first places the save start/end
*addresses in CBASSA and CBASEA, then calls the entry point below.
*If CBASSA is later found not to be zero, the prompt for the start,
*end and program execute addresses is not displayed.
*--------------------------------------------------------------------------
```

```
MAXFNLN EQU 20           Maximum file name length of 20 characters.
MAXFDLN EQU 70           Maximum program description length of 70 characters.

IDESV   DATA WSREG       BLWP vector for this command.
        DATA $+2

        CLR @SAVOVW      Clear save-overwrite flag.

*Get file name.

IDESV37 XOP @PNTFLN,14   Print prompt for file name.

        LI R1,MAXFNLN    Maximum name length.
        LI R3,'_'*256    Use underscore for text entry field.
        BL @TSENTRY      Accept text string for file name.

        JNE IDESV01      Jump if name length is not zero.

        RTWP             Name length is zero - abort command.

*Read sector 1 into buffer.

IDESV01 BL @RDSEC1       Read sector 1 into first sector buffer.

*Step through each word in sector 1. If the word is non-zero, it is a pointer
*to an FDR, so process that FDR.

        LI R3,IDEBUF1    Start address of sector buffer containing sector 1.
IDESV30 MOV *R3,*R3      Word in buffer zero?
        JNE IDESV33      No, check the file name in this FDR.

IDESV34 INCT R3          Point to the next word in the buffered sector 1.
        CI R3,IDEBUF1+512  Reached the end of the sector?
        JNE IDESV30      No, process next FDR.

        JMP IDESV35      Yes, continue further down the routine.

*                        (Number of sectors to read - 1 - already set up.)
IDESV33 MOV *R3,@IDEPM02  Set up sector number to read.
*                        (Command "read sectors with retry" already set up.)
        MOV @IDEBFA2,@IDEPM04  Read into second sector buffer.

        BL @IRWBUFF      Read from drive into sector buffer.

        CLR R2           Check if each character in the file name is the same as
IDESV31 CB @CLBUFF(R2),@IDEBUF2(R2)  in the buffered FDR.
        JNE IDESV34      Character is different, therefore not a duplicate file
*                        name in this FDR. Check the next FDR.
        INC R2
        CI R2,MAXFNLN
        JNE IDESV31

        XOP @PNTSV6,14   Duplicate file name found - prompt whether to overwrite.
IDESV39 XOP R2,11        Read and echo character from terminal into MSB of R2.
        CI R2,>4E00      'N'?
        JEQ IDESV37      Yes, prompt for another file name.
        CI R2,>5900      'Y'?
        JNE IDESV39      No, not 'Y' or 'N' - loop round and read another character.

        XOP @PNTCR,14    Print <CR><LF>.
        MOV R3,@SAVOVW   Save pointer to number of sector containing the FDR of the
*                        original file to delete. Will delete the file once the new
*                        file has been successfully written.

*Clear the FDR buffer ready for accepting the new program details.

IDESV35 LI R1,IDEBUF2    Address of sector buffer to use for FDR.
IDESV02 CLR *R1+         Fill FDR buffer with zeroes.
        CI R1,IDEBUF2+512
        JNE IDESV02

*Store file name in FDR buffer.

        CLR R1           Copy the file name entered to the FDR buffer.
IDESV04 MOVB @CLBUFF(R1),@IDEBUF2(R1)
        INC R1
        CI R1,MAXFNLN
        JNE IDESV04

*Get start, end and program execute addresses, if called from the IDE command scanner.
*If called from Cortex BASIC, get the start and end addresses passed in and set the
*program execution address to >FFFF to identify the file as a BASIC file.

        MOV @CBASSA,R4   Get BASIC start address.
        JEQ IDESV38      If zero, routine was called from IDE command scanner so prompt for addresses from user.
        MOV @CBASEA,R5   Get BASIC end address.
```

```
        SETO R10          Set program execution address to >FFFF.
        JMP IDESV51       Have the addresses, skip over prompts to user.

IDESV38 XOP @PNTSV1,14    Print prompt for start memory address.
        XOP R4,9          Read hexadecimal word from terminal into R4.
        DATA IDESV36      Exit if null entered.
        DATA IDESV36      Exit if non-hex entered.

        SRL R4,1          Round start memory address down to word boundary
        SLA R4,1          (so this is an even address).

        XOP @PNTSV2,14    Print prompt for end memory address.
        XOP R5,9          Read hexadecimal word from terminal into R5.
        DATA IDESV36      Exit if null entered.
        DATA IDESV36      Exit if non-hex entered.

        SRL R5,1          Round start memory address down to word boundary
        SLA R5,1          (so this is an even address).

        XOP @PNTSV3,14    Print prompt for program execute address.
        XOP R10,9         Read hexadecimal word from terminal into R10.
        DATA IDESV05      Use start memory address if null entered.
        DATA IDESV36      Exit if non-hex entered.

        JMP IDESV51       Skip next statement.

IDESV05 MOV R4,R10        Copy start memory address to program execute address.

*Store load (start), execute and end addresses in FDR buffer.

IDESV51 MOV R4,@IDEBUF2+MAXFNLN
        MOV R10,@IDEBUF2+MAXFNLN+2
        MOV R5,@IDEBUF2+MAXFNLN+4

        INC R5            Round end memory address up to word boundary (so this is an odd address).

*Calculate number of sectors required to store specified memory range, plus add 1 sector to store
*the FDR.

        MOV R5,R7         Copy end memory address.
        INC R7            Add one byte as the start and end memory addresses are
*                         both included in the file.
        S R4,R7           Subtract start memory address to give number of bytes to write - result in R7.

        MOV R7,@IDEBUF2+MAXFNLN+6  Store file size (in bytes) in FDR buffer while we have it.

        LI R8,512         Number of bytes per sector.
        CLR R6            R6 paired with R7 to form dividend.
        DIV R8,R6         Divide R6/R7 by 512. Integer result in R6, remainder in R7.
        MOV R7,R7         Is remainder zero?
        JEQ IDESV06       Yes, no remainder.
        INC R6            No, need another sector to store remainder.

IDESV06 INC R6            Need another sector to store the FDR.

*Read sector 0 into buffer to check if there is enough space free on the disk to store the file.

        BL @RDSEC0        Read sector 0 into first sector buffer.

        C R6,@IDEBUF1+MAXDNLN  Compare number of sectors needed with the number available.

        JLE IDESV07       Jump to continue if required number of sectors available.

        XOP @PNTSV4,14    Print error message.
        RTWP              Return.

*Get (optional) program description.

IDESV07 XOP @PNTSV5,14    Print prompt for program description.

        LI R1,MAXFDLN     Maximum program description length.
        LI R3,'_'*256     Use underscore for text entry field.
        BL @TSENTRY       Accept text string for program description.

*Store program description in FDR buffer.

        CLR R1            Copy the program description entered to the FDR buffer.
IDESV08 MOVB @CLBUFF(R1),@IDEBUF2+MAXFNLN+8(R1)
        INC R1
        CI R1,MAXFDLN
        JNE IDESV08

*Initialise a couple of variables.

        CLR R10           Used to store number of sector to store the FDR in.
        LI R2,IDEBUF2+MAXFNLN+8+MAXFDLN  Start address of list of sectors used in FDR.
```

```
*Find the first free sector in the sector bitmap in the cached sector 0.
*The first free sector number will be stored in R7.

IDESV10 LI R6,IDEBUF1+MAXDNLN+2  Address of the first word in the sector bitmap in the
*                                cached sector 0.
        LI R7,2          Number of sector corresponding to the first bit in the
*                        sector bitmap.

IDESV11 C *R6,@HFFFF     All sectors in this word of the sector bitmap in use?
        JEQ IDESV09      Yes, look at next word in the sector bitmap.

        LI R8,>8000      Identifies which bit in the word we're looking at.
*                        Start with the MS bit. Will use this later to set the bit
*                        position (free sector) actually used to 1.
        MOV *R6,R9       Get the word from the sector bitmap. (Leave the original word
*                        unchanged).
IDESV13 SLA R9,1         Shift the MS bit into the Carry status bit.
        JNC IDESV12      Jump if Carry status bit = 0 (that is, the sector corresponding
*                        to the bit last shifted is free).
        INC R7           Increment sector number for next bit we're going to look at.
        SRL R8,1         Shift the identifier to the bit we're going to look at.
        JMP IDESV13      Check next bit.

IDESV09 AI R7,16         Increment sector number for next word in sector bitmap.
        INCT R6          Address next word in the sector bitmap.
        JMP IDESV11      Check next word in the sector bitmap. A free sector should be
*                        found eventually as we checked there were enough free sectors earlier.

IDESV12 SOC R8,*R6       Set to 1 the bit position in the sector bitmap
*                        corresponding to the free sector found.

        DEC @IDEBUF1+MAXDNLN  Decrement number of sectors free in the cached sector 0.

*Reserve sector for storing the FDR in.

        MOV R10,R10      Reserved sector for FDR yet?
        JNE IDESV14      Yes, so jump and use this sector for file data.

        MOV R7,R10       Remember the sector number for the FDR. Will load sector 1, add
*                        the sector number and save again later.

        JMP IDESV10      Reserve another sector for the file data.

*Store the sector number found in the FDR.

IDESV14 MOV R7,*R2+

*Write data to the sector.
*R4=memory start address (or the address written from so far).
*R5=memory end address.

        LI R8,IDEBUF3    Use third sector buffer.
        LI R1,256        256 words to write to the sector.

*                        (Number of sectors to write - 1 - already set up.)
        MOV R7,@IDEPM02  Set up sector number to write.
        MOV @IDECMWR,@IDEPM03  Set up command "write sectors with retry".
        MOV @IDEBFA3,@IDEPM04  Write from third sector buffer.

IDESV17 MOV *R4+,*R8+    Copy word from memory to sector buffer.

        DEC R1           Decrement number of words to write for this sector.
        JNE IDESV18      Jump if haven't written all words for this sector.

        BL @IRWBUFF      Write from sector buffer to drive.

        C R4,R5          Reached memory end address (the data exactly filled the last sector)?
        JL IDESV10       No, find next free sector to use.
        JMP IDESV20      Yes, jump on to next part of routine.

IDESV18 C R4,R5          Reached memory end address?
        JL IDESV17       No, loop round to write another word.

        CLR R4           Write >0000s to rest of sector.
IDESV19 MOV R4,*R8+
        DEC R1
        JNE IDESV19

        BL @IRWBUFF      Write final sector from buffer to drive.

*Write cached sector 0 back to disk.

*                        (Number of sectors to read - 1 - already set up.)
IDESV20 CLR @IDEPM02     Set up sector number to write (sector 0).
*                        (Command "write sectors with retry" already set up.)
        MOV @IDEBFA1,@IDEPM04  Write from first sector buffer.
```

```
        BL @IRWBUFF      Write from sector buffer to drive.

*Write FDR to disk.

*                        (Number of sectors to read - 1 - already set up.)
        MOV R10,@IDEPM02  Set up sector number to write.
*                        (Command "write sectors with retry" already set up.)
        MOV @IDEBFA2,@IDEPM04  Write from second sector buffer.

        BL @IRWBUFF      Write from sector buffer to drive.

*Load sector 1, add the sector number where the FDR was stored, and write
*back to disk.

        BL @RDSEC1       Read sector 1 into first sector buffer.

        LI R6,IDEBUF1    Start address of first sector buffer.
IDESV15 MOV *R6,*R6      Word in buffer zero?
        JEQ IDESV16      Yes, jump.

        INCT R6          Address next word in sector buffer.
        JMP IDESV15      Check next word.

IDESV16 MOV R10,*R6      Add the sector where the FDR was stored to the list of sectors.

        MOV @IDECMWR,@IDEPM03  Set up command "write sectors with retry".

        BL @IRWBUFF      Write from sector buffer to drive.

*Are we overwriting an existing file, and now need to delete the original file?

        MOV @SAVOVW,R3   Get stored pointer to sector number of previous file's FDR.
        JEQ IDESV36      If no pointer was stored, exit.
*
*                        Now need to load previous file's FDR into second sector buffer so
*                        we can jump in part way through the Delete routine, after the point
*                        where the file name has been accepted and the file found.
*                        (Number of sectors to read - 1 - already set up.)
        MOV *R3,@IDEPM02  Set up sector number to read.
        MOV @IDECMRD,@IDEPM03  Set up command "read sectors with retry".
        MOV @IDEBFA2,@IDEPM04  Read into second sector buffer.

        BL @IRWBUFF      Read from drive into sector buffer.

        B @IDEDL08       Jump to the Delete routine, which will then RTWP to the
*                        calling routine - the command scanner or Cortex BASIC.

*All done!

IDESV36 XOP @PNTCR,14    Print <CR><LF>.
        RTWP             Return.

*--------------------------------------------------------------------------
*IDE DIR command.
*--------------------------------------------------------------------------
*Lists the name, load address, execute address, end address, size and
*program description of each file currently stored on the disk, plus the
*number of sectors free on the disk.

*Don't print the load, excute and end addresses for BASIC programs, which
*are identified by the execute address being >FFFF (which is invalid for
*machine-code programs).
*--------------------------------------------------------------------------

IDEDR   DATA WSREG       BLWP vector for this command.
        DATA $+2

        XOP @PNTCR,14    Print <CR><LF>.

*Read sector 1 into buffer.

        BL @RDSEC1       Read sector 1 into first sector buffer.

*Step through each word in sector 1. If the word is non-zero, it is a pointer
*to an FDR, so process that FDR.

        LI R3,IDEBUF1    Start address of sector buffer containing sector 1.
IDEDR01 MOV *R3,*R3      Word in buffer zero?
        JNE IDEDR03      No, process this FDR.

IDEDR06 INCT R3          Point to the next word in the buffered sector 1.
        CI R3,IDEBUF1+512  Reached the end of the sector?
        JNE IDEDR01      No, process next FDR.

*Read sector 0 from the drive and display the number of sectors free.

        BL @RDSEC0       Read sector 0 into first sector buffer.
```

```
        XOP @PNTPMT,14    Print '>' character.
        XOP @IDEBUF1+MAXDNLN,10 Print number of sectors free from sector buffer.
        XOP @PNTDR7,14    Print 'sectors free' and <CR><LF>.

        RTWP              Return.

*Load FDR and print file details.

*                         (Number of sectors to read - 1 - already set up.)
IDEDR03 MOV *R3,@IDEPM02  Set up sector number to read.
*                         (Command "read sectors with retry" already set up.)
        MOV @IDEBFA2,@IDEPM04  Read into second sector buffer.

        BL @IRWBUFF       Read from drive into sector buffer.

        CLR R2            Character counter for file name.
IDEDR02 CB @IDEBUF2(R2),@H00 Is character >00, which is non-printable?
        JEQ IDEDR04       Yes, jump to print a space character instead.
        XOP @IDEBUF2(R2),12  Print character from file name.
        JMP IDEDR05
IDEDR04 XOP @PNTDR1,12    Print space character.
IDEDR05 INC R2
        CI R2,MAXFNLN
        JNE IDEDR02

        C @IDEBUF2+MAXFNLN+2,@HFFFF  Is execute address >FFFF?
        JNE IDEDR07       No, print load, execute and end addresses.
        XOP @PNTDR8,14    Yes, print "BASIC program" instead of addresses.
        JMP IDEDR08       Print file size details.

IDEDR07 XOP @PNTDR1,14    Print Load address intro text.
        XOP @IDEBUF2+MAXFNLN,10  Print Load address.
        XOP @PNTDR2,14    Print Execute address intro text.
        XOP @IDEBUF2+MAXFNLN+2,10  Print execute address.
        XOP @PNTDR3,14    Print End address intro text.
        XOP @IDEBUF2+MAXFNLN+4,10  Print end address.
IDEDR08 XOP @PNTDR4,14    Print Size intro text.
        XOP @IDEBUF2+MAXFNLN+6,10  Print size.
        XOP @PNTDR5,14    Print 'bytes', <CR><LF> and indent for the program description
*                         on the next line.

        CLR R2            Character counter for program description.
IDERP07 CB @IDEBUF2+MAXFNLN+8(R2),@H00  Is character >00, which is non-printable?
        JEQ IDERP08       Yes, jump to print a space character instead.
        XOP @IDEBUF2+MAXFNLN+8(R2),12  Print character from file name.
        JMP IDERP09
IDERP08 XOP @PNTDR1,12    Print space character.
IDERP09 INC R2
        CI R2,MAXFDLN
        JNE IDERP07

        XOP @PNTDR6,14    Print closing bracket and <CR><LF>.

        JMP IDEDR06       Return.

*---------------------------------------------------------------------------
*IDE LOAD command.
*---------------------------------------------------------------------------
*Requests file name to load from the user. Checks that the specified
*file exists, and gets the load address, execute address and file size
*from the FDR. Then loads from disk to memory and returns to the prompt.

*When called from Cortex BASIC, BASIC first places the load start address
*in CBASSA, then calls the entry point below. If CBASSA is
*later found not to be zero, the program is loaded at the address in CBASSA
*rather than at the save/load address saved in the file. (BASIC programs
*are relocatable relative to the beginning of the user storage area.)
*---------------------------------------------------------------------------

IDELD   DATA WSREG        BLWP vector for this command.
        DATA $+2

*Get file name.

        CLR R9            Set status value to indicate IDE LOAD command (as
*                         opposed to IDE RUN command).

IDELD06 XOP @PNTFLN,14    Print prompt for file name.

        LI R1,MAXFNLN     Maximum name length.
        LI R3,'_'*256     Use underscore for text entry field.
        BL @TSENTRY       Accept text string for file name.

        JNE IDELD01       Jump if name length is not zero.

        RTWP              Name length is zero - abort command.
```

```
*Check if the specified file name exists.

IDELD01 LI R10,IDELD06   Address to jump to to request another file name if the specified
*                        name is not found.

        BL @FNEXSTS      Check if specified file name exists.

*Have found the specified file. Its FDR is still cached in the second sector buffer.
*Get the load address, execute address and file size.

        MOV @IDEBUF2+MAXFNLN, R4  Get the load address.
        MOV @IDEBUF2+MAXFNLN+2,R10  Get the execute address.
        MOV @IDEBUF2+MAXFNLN+6,R7  Get the file size.

*If called from Cortex BASIC, replace the load start address from the file with the load
*start address passed in.

        MOV @CBASSA,R2   Get BASIC start address.
        JEQ IDELD38      If zero, routine was called from IDE command scanner, so jump.
        MOV R2,R4        Replace load address with BASIC start address.

*Set pointer to the list of data sectors used in the FDR.

IDELD38 LI R2,IDEBUF2+MAXFNLN+8+MAXFDLN

*Read data sector.

*                        (Number of sectors to read - 1 - already set up.)
IDELD04 MOV *R2+,@IDEPM02  Set up sector number to read.
*                        (Command "read sectors with retry" already set up.)
        MOV @IDEBFA1,@IDEPM04  Read into first sector buffer.

        BL @IRWBUFF      Read from drive into sector buffer.

        LI R8,IDEBUF1    Start address of the first sector buffer.
        LI R1,256        256 words to read from the sector.

*Copy data from sector buffer to memory.

IDELD02 MOV *R8+,*R4+    Copy word from sector buffer to memory.

        DECT R7          Decrement file size by one word. Read required number of bytes?
        JEQ IDELD03      Yes, exit loop.

        DEC R1           Decrement number of words to read from this sector.
        JNE IDELD02      Jump if haven't read all words from this sector.

        JMP IDELD04      Read all words from this sector. Read next sector.

IDELD03 DECT R4          Decrement R4 back to the address of the last word loaded.
        MOV R4,@CBASEA   Pass end load address back to BASIC.

        MOV R9,R9        Status value set to zero (for the IDE LOAD command)?
        JEQ IDELD05      Yes, jump to return to command prompt.

        B *R10           Run program from program execute address.

IDELD05 XOP @PNTCR,14    Print <CR><LF>.
        RTWP             Return.

*--------------------------------------------------------------------------
*IDE RUN command.
*--------------------------------------------------------------------------
*As the IDE LOAD command, but runs the program from the execute address on
*completion of loading.
*--------------------------------------------------------------------------

IDERN   DATA WSREG       BLWP vector for this command.
        DATA $+2

        SETO R9          Set status value to indicate IDE RUN command (as
*                        opposed to IDE LOAD command).

        JMP IDELD06      Jump to IDE LOAD command to handle everything else.

*--------------------------------------------------------------------------
*IDE RENAME command.
*--------------------------------------------------------------------------
*Renames the specified file.
*--------------------------------------------------------------------------

IDERNF  DATA WSREG       BLWP vector for this command.
        DATA $+2

IDERF04 XOP @PNTFLN,14   Print prompt for file name.
```

```
        LI R1,MAXFNLN    Maximum name length.
        LI R3,'_'*256    Use underscore for text entry field.
        BL @TSENTRY      Accept text string for file name.

        JNE IDERF01      Jump if name length is not zero.

        RTWP             Name length is zero - abort command.

*Check if the specified file name exists.

IDERF01 LI R10,IDERF04   Address to jump to to request another file name if the specified
*                        name is not found.

        BL @FNEXSTS      Check if specified file name exists.

*Have found the specified file. Its FDR is still cached in the second sector buffer.
*Its sector number is in *R3.

*Get new file name.

        XOP @PNTNFLN,14  Print prompt for new file name.

        LI R1,MAXFNLN    Maximum name length.
        LI R3,'_'*256    Use underscore for text entry field.
        BL @TSENTRY      Accept text string for file name.

        JNE IDERF02      Jump if name length is not zero.

        RTWP             Name length is zero - abort command.

*Store the new file name in FDR buffer.

IDERF02 CLR R1           Copy the new file name entered to the FDR buffer.
IDERF03 MOVB @CLBUFF(R1),@IDEBUF2(R1)
        INC R1
        CI R1,MAXFNLN
        JNE IDERF03

*Write the FDR sector back to the disk.

*                        (Parameters already set up for FDR sector)
        MOV @IDECMWR,@IDEPM03  Set up command "write sectors with retry".
        BL @IRWBUFF      Write from sector buffer to drive.

*All done!

        XOP @PNTCR,14    Print <CR><LF>.
        RTWP             Return.

*--------------------------------------------------------------------------------
*IDE PROGRAM DESCRIPTION command.
*--------------------------------------------------------------------------------
*Changes the (optional) program description for the specified file.
*--------------------------------------------------------------------------------

IDERPD  DATA WSREG       BLWP vector for this command.
        DATA $+2

IDERP10 XOP @PNTFLN,14   Print prompt for file name.

        LI R1,MAXFNLN    Maximum name length.
        LI R3,'_'*256    Use underscore for text entry field.
        BL @TSENTRY      Accept text string for file name.

        JNE IDERP01      Jump if name length is not zero.

        RTWP             Name length is zero - abort command.

*Check if the specified file name exists.

IDERP01 LI R10,IDERP10   Address to jump to to request another file name if the specified
*                        name is not found.

        BL @FNEXSTS      Check if specified file name exists.

*Have found the specified file. Its FDR is still cached in the second sector buffer.
*Its sector number is in *R3.

*Display the current program description.

        XOP @PNTFPD,14   Print program description heading.

        CLR R2           Character counter.
IDERP02 CB @IDEBUF2+MAXFNLN+8(R2),@H00 Is character >00, which is non-printable?
        JEQ IDERP04      Yes, jump to print a space character instead.
        XOP @IDEBUF2+MAXFNLN+8(R2),12  Print character from file name.
        JMP IDERP05
```

```
IDERP04 XOP @PNTDR1,12   Print space character.
IDERP05 INC R2
        CI R2,MAXFDLN
        JNE IDERP02

*Get new program description.

        XOP @PNTNFPD,14   Print prompt for new program description.

        LI R1,MAXFDLN    Maximum program description length.
        LI R3,'_'*256    Use underscore for text entry field.
        BL @TSENTRY      Accept text string for program description.

*Store new program description in FDR buffer.

        CLR R1           Copy the program description entered to the FDR buffer.
IDERP06 MOVB @CLBUFF(R1),@IDEBUF2+MAXFNLN+8(R1)
        INC R1
        CI R1,MAXFDLN
        JNE IDERP06

*Write the FDR sector back to the disk.

*                        (Parameters already set up for FDR sector)
        MOV @IDECMWR,@IDEPM03  Set up command "write sectors with retry".
        BL @IRWBUFF      Write from sector buffer to drive.

*All done!

        RTWP             Return.

*--------------------------------------------------------------------------
*IDE DELETE command.
*--------------------------------------------------------------------------
*Deletes the specified file.
*--------------------------------------------------------------------------

IDEDL   DATA WSREG       BLWP vector for this command.
        DATA $+2

IDEDL07 XOP @PNTFLN,14   Print prompt for file name.

        LI R1,MAXFNLN    Maximum name length.
        LI R3,'_'*256    Use underscore for text entry field.
        BL @TSENTRY      Accept text string for file name.

        JNE IDEDL01      Jump if name length is not zero.

        RTWP             Name length is zero - abort command.

*Check if the specified file name exists.

IDEDL01 LI R10,IDEDL07   Address to jump to to request another file name if the specified
*                        name is not found.

        BL @FNEXSTS      Check if specified file name exists.

*Have found the specified file. Its FDR is still cached in the second sector buffer.
*Its sector number is in *R3.

*Clear the pointer to the FDR in sector 1 (first sector buffer), and write sector 1 back to disk.

IDEDL08 MOV *R3,R4       Copy the FDR sector number - need it later.
        CLR *R3          Clear the pointer to the FDR in sector 1.

*                        (Number of sectors to write - 1 - already set up.)
        MOV @H0001,@IDEPM02  Set up sector number to write (sector 1).
        MOV @IDECMWR,@IDEPM03  Set up command "write sectors with retry".
        MOV @IDEBFA1,@IDEPM04  Write from first sector buffer.

        BL @IRWBUFF      Write from sector buffer to drive.

*Mark the FDR sector plus all the data sectors used by the file as free in the
*sector 0 bitmap. Start with the FDR sector, then loop through the list of data sectors
*used in the FDR. For each sector marked as free, increment the number of sectors free count.

        BL @RDSEC0       Read sector 0 into first sector buffer.

        LI R2,IDEBUF2+MAXFNLN+8+MAXFDLN-2  Address of word before list of data sectors used
*                        in the FDR.

IDEDL06 AI R4,-2         Subtract number of sector corresponding to the first bit in the
*                        sector bitmap (first bit in the sector bitmap is for sector 2).

        LI R5,16         16 sectors per word in the sector bitmap.
        CLR R3           R3 paired with R4 to form dividend.
        DIV R5,R3        Divide R3/R4 by 16. Integer result in R3 remainder in R4.
```

```
*                       So R3 is the number of words we need to move into the sector
*                       bitmap, and R4 is the bit position (from the left) in that word
*                       that relates to the FDR sector.

        SLA R3,1        Multiply R3 by 2 to convert to words.
        AI R3,IDEBUF1+MAXDNLN+2  Add to address of start of sector bitmap to address the word
*                       in the sector bitmap relating to the FDR sector.

        LI R5,>8000     Start with the left-most bit of the word selected.
        MOV R4,R4       Was there a remainder?
        JEQ IDEDL05     If no, exit the loop - bit position already set correctly in R5.
IDEDL04 SRL R5,1
        DEC R4
        JNE IDEDL04

IDEDL05 SZC R5,*R3      Reset the bit in the sector bitmap as indicated by the mask in R5.
        INC @IDEBUF1+MAXDNLN  Increment the number of sectors free count.

        INCT R2         Point to next sector number in list of sectors in the FDR.
        MOV *R2,R4      Get next sector number. Is it zero?
        JNE IDEDL06     No, loop round and process sector number.

*Write the sector 0 bitmap back to the disk.

*                       (Parameters already set up for sector 0)
        MOV @IDECMWR,@IDEPM03  Set up command "write sectors with retry".
        BL @IRWBUFF     Write from sector buffer to drive.

*All done!

        RTWP            Return.

*-------------------------------------------------------------------------------
*IDE FORMAT DISK command.
*-------------------------------------------------------------------------------
*Requests a disk name from the user, then formats the disk by writing
*zeroes to all sectors, then writing the disk name and number of free
*sectors to the start of sector 0.
*-------------------------------------------------------------------------------

MAXDNLN EQU 20          Maximum disk name length of 20 characters.

IDEFM   DATA WSREG      BLWP vector for this command.
        DATA $+2

        XOP @PNTDSKN,14  Print prompt for disk name.

        LI R1,MAXDNLN   Maximum name length.
        LI R3,>5F00     Use underscore for text entry field.
        BL @TSENTRY     Accept text string for disk name.

        JNE IDEFM1      Jump if name length is not zero.

        RTWP            Name length is zero - abort command.

IDEFM1  LI R0,IDEBUF1   Address of sector buffer to use.
IDEFM6  CLR *R0+        Fill sector buffer with zeroes.
        CI R0,IDEBUF1+512
        JNE IDEFM6

        LI R4,512-MAXDNLN-2*8-1  Total number of sectors supported.
*                       Limited by number of sectors supported in the sector bitmap in sector 0.
*                       512 bytes minus 20 bytes for the disk name, minus another two bytes for
*                       the number of free sectors, times 8 sectors per byte.
*                       Subtract a further 1 as sectors are numbered from zero.

        CLR R2          Counter for current sector.

        XOP @PNTDSK1,14  Print initial progress message, followed by ...
        XOP R4,10       ... the total number of sectors supported.

        LI R9,12        Cursor left back 12 characters to the start of the current sector number.
IDEFM2  XOP @PNTCRL,14
        DEC R9
        JNE IDEFM2

        MOV @H0100,@IDEPM01  Set up number of sectors to write at a time in MS byte.
IDEFM5  MOV R2,@IDEPM02  Set up sector number to format.
        MOV @IDECMWR,@IDEPM03  Set up command "write sectors with retry".
        MOV @IDEBFA1,@IDEPM04  Write from first sector buffer.

        BL @IRWBUFF     Write from sector buffer to drive.

        INC R2          Increment current sector number.
        C R2,R4         Done all sectors?
        JGT IDEFM3      Yes, jump.
```

```
        XOP R2,10        Print (incremented) current sector number.
        LI R9,4          Cursor left back 4 characters to the start of the current sector number.
IDEFM4  XOP @PNTCRL,14
        DEC R9
        JNE IDEFM4

        JMP IDEFM5       Jump to clear next sector.

IDEFM3  XOP @PNTCR,14    Print <CR><LF>.

        LI R0,512-MAXDNLN-2*8  Total number of sectors supported.
        MOV R0,@IDEBUF1+MAXDNLN  Copy number of sectors supported to sector buffer after disk name.

IDEFM8  CLR R2           Copy the disk name entered to the start of the sector buffer.
*                        (Do this AFTER writing the number of sectors supported because this section of code
*                         is also used by the IDE RENAME DISK routine, where the name of the disk needs to be
*                         changed without changing the number of sectors currently free.)
IDEFM7  MOVB @CLBUFF(R2),@IDEBUF1(R2)
        INC R2
        CI R2,MAXDNLN
        JNE IDEFM7

        CLR @IDEPM02     Set up sector number to write (sector 0).
        MOV @IDECMWR,@IDEPM03  Set up command "write sectors with retry" (was set to read sectors from the
*                              IDE RENAME DISK routine).

        BL @IRWBUFF      Write from sector buffer to drive.

        RTWP             Resturn.

*--------------------------------------------------------------------------
*IDE RENAME DISK command.
*--------------------------------------------------------------------------
*Requests a disk name from the user, then reads sector 0, replaces the old
*disk name with the new name, and writes sector 0 back to the disk.
*--------------------------------------------------------------------------

IDERM   DATA WSREG       BLWP vector for this command.
        DATA $+2

        XOP @PNTDSKN,14  Print prompt for disk name.

        LI R1,MAXDNLN    Maximum name length.
        LI R3,>5F00      Use underscore for text entry field.
        BL @TSENTRY      Accept text string for disk name.

        JNE IDERM1       Jump if name length is not zero.

        RTWP             Name length is zero - abort command.

IDERM1  BL @RDSEC0       Read sector 0 into first sector buffer.

        JMP IDEFM8       Replace name and write back to disk - use the code in the
*                        IDE FORMAT DISK routine.

**************************************************************************
*Common subroutine:                                                      *
*Read sector 0 into first sector buffer.                                 *
**************************************************************************

RDSEC0  MOV R11,@CSBTRN1  Save return address.

        MOV @H0100,@IDEPM01  Set up number of sectors to read in MS byte.
        CLR @IDEPM02     Set up sector number to read (sector 0).
        MOV @IDECMRD,@IDEPM03  Set up command "read sectors with retry".
        MOV @IDEBFA1,@IDEPM04  Read into first sector buffer.

        BL @IRWBUFF      Read from drive into sector buffer.

        MOV @CSBTRN1,R11  Restore return address.
        B *R11           Return.

**************************************************************************
*Common subroutine:                                                      *
*Read sector 1 into first sector buffer.                                 *
**************************************************************************

RDSEC1  MOV R11,@CSBTRN1  Save return address.

        MOV @H0100,@IDEPM01  Set up number of sectors to read in MS byte.
        MOV @H0001,@IDEPM02  Set up sector number to read (sector 1).
        MOV @IDECMRD,@IDEPM03  Set up command "read sectors with retry".
        MOV @IDEBFA1,@IDEPM04  Read into first sector buffer.

        BL @IRWBUFF      Read from drive into sector buffer.

        MOV @CSBTRN1,R11  Restore return address.
```

```
        B *R11          Return.

*******************************************************************************
*Common subroutine:                                                           *
*Find if specified file name exists.                                          *
*Step through each word in sector 1. If the word is non-zero, it is a         *
*pointer to an FDR, so check if that FDR contains the specified file name. *
*******************************************************************************
*The name of the file to find must be in the CLBUFF array.
*If the file name is *not* found, a 'file not found' error message is
*displayed and the code jumps to the address which must have previously
*been set up in R10.
*If the file name *is* found, the routine exits with the file's FDR cached
*in the second sector buffer, and its FDR sector number in *R3.

FNEXSTS MOV R11,@CSBTRN2  Save return address.

        BL @RDSEC1        Read sector 1 into first sector buffer.

        LI R3,IDEBUF1     Start address of sector buffer containing sector 1.
FNEXS01 MOV *R3,*R3       Word in buffer zero?
        JNE FNEXS03       No, check the file name in this FDR.

FNEXS02 INCT R3           Point to the next word in the buffered sector 1.
        CI R3,IDEBUF1+512  Reached the end of the sector?
        JNE FNEXS01       No, process next FDR.

        XOP @PNTLD1,14    Specified file name not found - print error message.
        B *R10            Get another file name.

*                         (Number of sectors to read - 1 - already set up.)
FNEXS03 MOV *R3,@IDEPM02   Set up sector number to read.
*                         (Command "read sectors with retry" already set up.)
        MOV @IDEBFA2,@IDEPM04  Read into second sector buffer.

        BL @IRWBUFF       Read from drive into sector buffer.

        CLR R2            Check if each character in the file name is the same as
FNEXS04 CB @CLBUFF(R2),@IDEBUF2(R2)  in the buffered FDR.
        JNE FNEXS02       Character is different, therefore not the specified
*                         file name. Check the next FDR.
        INC R2
        CI R2,MAXFNLN
        JNE FNEXS04

        MOV @CSBTRN2,R11  Restore return address.
        B *R11            Return.

*******************************************************************************
*IDE subroutine:                                                              *
*Wait until drive is ready and not busy.                                      *
* NEED TO ADD COMMAND/DEVICE ERROR CHECKING TO ALL ROUTINES AT SOME POINT.
*******************************************************************************
*Inputs:                                                                      *
*- None -                                                                     *
*-----------------------------------------------------------------------------*
*Outputs:                                                                     *
*- None -                                                                     *
*-----------------------------------------------------------------------------*
*Uses:                                                                        *
*R1: destroyed.                                                               *
*R11: return address.                                                         *
*******************************************************************************

IWTRDY  MOV @IDERR7,R1    Read status from IDE register 7.
        COC @ISTRDY,R1    Check if "Ready" status bit is set.
        JNE IWTRDY        Bit not set yet. Loop until it is set.

        COC @ISTBSY,R1    Check if "Busy" status bit is set.
        JEQ IWTRDY        Bit set. Loop until it is clear.

        B *R11            Return.

*******************************************************************************
*IDE subroutine:                                                              *
*Write sector number to IDE LBA registers.                                    *
*******************************************************************************
*Inputs:                                                                      *
*IDEPM02: sector number to write to IDE LBA registers.                        *
*-----------------------------------------------------------------------------*
*Outputs:                                                                     *
*- None -                                                                     *
*-----------------------------------------------------------------------------*
*Uses:                                                                        *
*R1: destroyed.                                                               *
*R11: return address.                                                         *
*******************************************************************************
```

```
ILBAA   SWPB @IDEPM02    Move LS byte of sector number into MS byte.
        MOV @IDEPM02,@IDEWR3  Write to IDE register 3 (sector address LBA 0).
        SWPB @IDEPM02    Swap bytes back round the right way again - need to keep
*                        sector number intact.

        MOV @IDEPM02,@IDEWR4  Write MS byte of sector number to IDE register 4 (sector address LBA 1).

        CLR @IDEWR5      Clear IDE register 5 (sector address LBA 2).

        MOV @IDEMD,@IDEWR6  Select master drive (and sector address LBA 3).

        B *R11           Return.

********************************************************************************
*IDE subroutine:                                                               *
*Read from drive into sector buffer, or write from drive into sector           *
*buffer. The direction is determined by the IDE command specified in the       *
*input parameters.                                                             *
********************************************************************************
*Inputs:                                                                       *
*IDE parameters below to be set up before calling this routine.                *
*------------------------------------------------------------------------------*
*Outputs:                                                                      *
*- None -                                                                      *
*------------------------------------------------------------------------------*
*Uses:                                                                         *
*R1: destroyed.                                                                *
*R11: return address.                                                          *
********************************************************************************

IRWBUFF MOV R11,@IDERTN  Save return address.

        BL @IWTRDY       Wait until drive is ready and not busy.

        MOV @IDEPM01,@IDEWR2  Write number of sectors to read to IDE register 2.

        BL @ILBAA        Write the sector number to read to IDE LBA registers.

        MOV @IDEPM03,@IDEWR7   Write command to IDE register 7.

IRWBUF1 MOV @IDERR7,R1   Read status from IDE register 7.
        COC @ISTBSY,R1   Check if "Busy" status bit is set.
        JEQ IRWBUF1      Bit set. Loop until it is clear.

        COC @ISTDRQ,R1   Check if "Data Request Ready" status bit is set.
        JNE IRWBUF1      Bit not set yet. Loop until it is set.

        MOV @IDEPM04,R1  Get start address of sector buffer.
        MOV R1,@IDEPM05  Calculate end address of sector buffer.
        A @H0200,@IDEPM05

        C @IDEPM03,@IDECMWR  Is IDE command a write operation?
        JEQ IRWBUF3      Yes, jump to write routine.
*                        (No, so must be a read operation or drive ID operation.)

IRWBUF2 MOV @IDERR0,*R1+  Copy word from drive to sector buffer.
        C R1,@IDEPM05    Filled buffer?
        JNE IRWBUF2      No, loop round.

        JMP IRWBUF5      Jump to exit.

IRWBUF3 MOV *R1+,@IDEWR0  Copy word from sector buffer to drive.
        C R1,@IDEPM05    Filled buffer?
        JNE IRWBUF3      No, loop round.

IRWBUF4 MOV @IDERR7,R1   Read status from IDE register 7.
        COC @ISTBSY,R1   Check if "Busy" status bit is set.
        JEQ IRWBUF4      Bit set. Loop until it is cleared after sector written to disk.

IRWBUF5 MOV @IDERTN,R11  Restore return address.
        B *R11           Return.

********************************************************************************
*Text string entry subroutine.                                                 *
********************************************************************************
*Allows the user to input a text string up to a maximum specified length.      *
*The text string can be editted during entry using the <Backspace> button.     *
*If the maximum entry length is reached, further text entry just replaces      *
*the last character in the string. A visible text entry field can be           *
*displayed.                                                                    *
*------------------------------------------------------------------------------*
*Inputs:                                                                       *
*R1: maximum length of text string allowed.                                    *
*R3: MSB contains ASCII code of character to use to display text entry         *
*    field. Use >5F00 for underscore. Use >2000 for space (i.e. no visible     *
*    text entry field).                                                        *
```

```
*--------------------------------------------------------------------------*
*Outputs:                                                                  *
*R1: actual length of text string entered.                                 *
*CLBUFF: text string stored in this entry buffer.                          *
*--------------------------------------------------------------------------*
*Uses:                                                                     *
*R2: destroyed.                                                            *
*R10: destroyed.                                                            *
*R11: return address.                                                      *
****************************************************************************

TSENTRY CLR R2           Clear specified number of characters in entry buffer.
TSENT09 MOVB @H00,@CLBUFF(R2)
        INC R2
        C R2,R1
        JNE TSENT09

        MOV R1,R2        Copy maximum length allowed value.
TSENT01 XOP R3,12        Print characters to display text entry field ...
        DEC R2
        JNE TSENT01

        MOV R1,R2        ... then cursor left back over them to the first character position.
TSENT02 XOP @PNTCRL,14
        DEC R2           R2 finally left at zero ready to hold actual length of text string entered.
        JNE TSENT02

TSENT03 XOP R10,13       Read character from terminal into R10.

        CI R10,>0D00     Was character a <CR>?
        JEQ TSENT99      Yes, exit routine.

        CI R10,>1B00     Was character an <Escape>?
        JNE TSENT10      No, continue.
        CLR R2           Yes, set number of characters to zero.
        JMP TSENT99      Exit routine.

TSENT10 CI R10,>0800     Was character a backspace?
        JNE TSENT04      No, print character and store in buffer.

        MOV R2,R2        Is text string entry length currently zero?
        JEQ TSENT03      Yes, ignore backspace character and read next character.

        C R2,R1          Was maximum number of characters already entered, in which case cursor
*                        is already positioned over the last character?
        JEQ TSENT07      Yes, jump to skip next instruction.

        XOP R10,12       Print the backspace character (moves cursor over last character).

TSENT07 XOP R3,12        Print text entry field display character over last character.
        XOP @PNTCRL,14   Cursor left back over text entry field display character.
        MOVB @H00,@CLBUFF-1(R2)  Clear character in buffer.
        DEC R2           Decrement text string entry length.
        JMP TSENT03      Read next character.

TSENT04 XOP R10,12       Print character.
        INC R2           Increment text string entry length.

        C R2,R1          Compare number of characters now entered with maximum length allowed.
        JLT TSENT08      Haven't reached maximum. Just store character in buffer.
        JEQ TSENT06      Have now reached maximum. Cursor now positioned after the entry field.
*                        Move character back over last character then store character in buffer.

        DEC R2           Were already at maximum entry length. Decrement text string entry length
*                        back to what it was as the last character has been replaced rather than a
*                        new character added.

TSENT06 XOP @PNTCRL,14   Move cursor back over last character.

TSENT08 MOVB R10,@CLBUFF-1(R2)  Store character in buffer.
        JMP TSENT03      Read next character.

TSENT99 XOP @PNTCR,14    Print <CR><LF>.

        MOV R2,R1        Move text string entry length to R1. Status EQ bit set if name length is
*                        zero, so name length can be checked immediately on returning.

        B *R11           Return.

*--------------------------------------------------------------------------

IDEBASE EQU >E000        Base address in memory map for IDE interface registers.
IDERR0  EQU IDEBASE      IDE interface Register 0 read address.
IDERR7  EQU IDEBASE+>0E  IDE interface Register 7 read address.
IDEWR0  EQU IDEBASE+>10  IDE interface Register 0 write address.
IDEWR2  EQU IDEBASE+>14  IDE interface Register 2 write address.
IDEWR3  EQU IDEBASE+>16  IDE interface Register 3 write address.
```

```
IDEWR4  EQU IDEBASE+>18  IDE interface Register 4 write address.
IDEWR5  EQU IDEBASE+>1A  IDE interface Register 5 write address.
IDEWR6  EQU IDEBASE+>1C  IDE interface Register 6 write address.
IDEWR7  EQU IDEBASE+>1E  IDE interface Register 7 write address.

ISTRDY  DATA >4000       IDE interface "Ready" status bit.
ISTBSY  DATA >8000       IDE interface "Busy" status bit.
ISTDRQ  DATA >0800       IDE interface "Data Request Ready" status bit.

IDEMD   DATA >E000       IDE register 6 value to select Master drive.
*                        Also IDE command "Power down drive".
IDECMPU DATA >E100       IDE command "Power up drive".
IDECMID DATA >EC00       IDE command "Drive ID".
IDECMRD DATA >2000       IDE command "read sectors with retry".
IDECMWR DATA >3000       IDE command "write sectors with retry".

H0001   DATA >0001       Hex value >0001.
H0100   DATA >0100       Hex value >0100 (256 decimal).
H0200   DATA >0200       Hex value >0200 (512 decimal).
HFFFF   DATA >FFFF       Hex value >FFFF.

IDEBFA1 DATA IDEBUF1     Address of first sector buffer.
IDEBFA2 DATA IDEBUF2     Address of second sector buffer.
IDEBFA3 DATA IDEBUF3     Address of third sector buffer.

CLSCRN  DATA >1B5B,>324A  VT100 escape code to erase screen.
        BYTE >1B,>5B,>48  VT100 escape code to move cursor to top left.
H00     BYTE 0           Used for hex 0 byte value.

WELCME  TEXT 'TM 990 DISK OPERATING SYSTEM'
        BYTE >0D,>0A
        TEXT 'Version 0.1'
        BYTE 0

PNTCR2  BYTE >0D,>0A
PNTCR   BYTE >0D,>0A
        BYTE 0

PNTPMT  TEXT '>'         Command prompt.
        BYTE 0

PNTCRL  BYTE >1B,>5B,>44  VT100 escape code to move cursor one character left.
        BYTE 0

PNTERR1 TEXT 'ERROR: command not found'
        BYTE >0D,>0A
        BYTE 0

PNTCSP1 BYTE >0D,>0A
        TEXT 'Commands supported: '
        BYTE 0

PNTCSP2 BYTE >0D,>0A
        TEXT '                    '
        BYTE 0

PNTDID1 BYTE >0D,>0A
        TEXT 'IDE drive name:              '  Last two spaces also used as a text string.
        BYTE 0

PNTDID2 BYTE >0D,>0A
        TEXT 'Number of sectors on drive:  >'
        BYTE 0

PNTDID3 BYTE >0D,>0A
        TEXT 'Disk name:                   '
        BYTE 0

PNTDID4 BYTE >0D,>0A
        TEXT 'Number of sectors supported: >'
        BYTE 0

PNTDID5 BYTE >0D,>0A
        TEXT 'Number of sectors free:      >'
        BYTE 0

PNTIST  DATA PNTIST1     Address table for text for each status bit.
        DATA PNTIST2
        DATA PNTIST3
        DATA PNTIST4
        DATA PNTIST5
        DATA PNTIST6
        DATA PNTIST7
        DATA PNTIST8

PNTIST1 BYTE >0D,>0A
        TEXT 'Bit 7 - BSY = '
```

```
        BYTE 0
PNTIST2 TEXT '    Bit 6 - RDY = '
        BYTE 0
PNTIST3 BYTE >0D,>0A
        TEXT 'Bit 5 - DF  = '
        BYTE 0
PNTIST4 TEXT '    Bit 4 - DSC = '
        BYTE 0
PNTIST5 BYTE >0D,>0A
        TEXT 'Bit 3 - DRQ = '
        BYTE 0
PNTIST6 TEXT '    Bit 2 - COR = '
        BYTE 0
PNTIST7 BYTE >0D,>0A
        TEXT 'Bit 1 - IDX = '
        BYTE 0
PNTIST8 TEXT '    Bit 0 - ERR = '
        BYTE 0

PNTGSN1 BYTE >0D,>0A
        TEXT 'Number of sector to view: >'  Colon also used as a single character.
        BYTE 0

PNTGSN2 BYTE >0D,>0A
        TEXT 'Contents of sector >'
        BYTE 0

PNTGSN3 TEXT ' ADDR  DATA----------------------------------  ASCII-----------'
        BYTE >0D,>0A
        BYTE 0

PNTGSN4 BYTE >0D,>0A
        TEXT 'Press: (P)revious sector, (N)ext sector, (Q)uit '
        BYTE 0

PNTGSF1 BYTE >0D,>0A
        TEXT 'Start sector number: >'
        BYTE 0

PNTGSF2 BYTE >0D,>0A
        TEXT 'Number of sectors to fill: >'
        BYTE 0

PNTGSF3 BYTE >0D,>0A
        TEXT 'Word value to fill with: >'
        BYTE 0

PNTSV1  BYTE >0D,>0A
        TEXT 'Save from memory address: >'
        BYTE 0

PNTSV2  BYTE >1B,>5B,>33,>31,>43  VT100 escape code to move cursor 31 characters right
*                                 (to end of previous message).
        TEXT ' to memory address (inclusive): >'
        BYTE 0

PNTSV3  BYTE >0D,>0A
        TEXT 'Program execute address (defaults to save start address): >'
        BYTE 0

PNTSV4  BYTE >0D,>0A
        BYTE >0D,>0A
        TEXT '*** Not enough space on disk to store specified memory range ***'
        BYTE 0

PNTSV5  BYTE >0D,>0A
        BYTE >0D,>0A
        TEXT 'Program description (optional):'
        BYTE >0D,>0A
        BYTE 0

PNTSV6  BYTE >0D,>0A
        TEXT 'Overwrite existing file (Y/N)? '
        BYTE 0

PNTDR1  TEXT ' (load: >'
        BYTE 0

PNTDR2  TEXT ', exec: >'
        BYTE 0

PNTDR3  TEXT ', end: >'
        BYTE 0

PNTDR4  TEXT ', size: >'
        BYTE 0
```

```
PNTDR5  TEXT ' bytes)'
        BYTE >0D,>0A
        TEXT '      ['
        BYTE 0

PNTDR6  TEXT ']'
        BYTE >0D,>0A
        BYTE 0

PNTDR7  TEXT ' sectors free'
        BYTE >0D,>0A
        BYTE 0

PNTDR8  TEXT ' ( ---------- BASIC program ----------'
        BYTE 0

PNTLD1 BYTE >0D,>0A
        TEXT '*** Specified file not found ***'
        BYTE >0D,>0A
        BYTE 0

PNTDSKN BYTE >0D,>0A
        TEXT 'Disk name: '
        BYTE 0

PNTDSK1 BYTE >0D,>0A
        TEXT 'Formatting sector number >0000 / >'
        BYTE 0

PNTFLN  BYTE >0D,>0A
        TEXT 'File name: '
        BYTE 0

PNTNFLN BYTE >0D,>0A
        TEXT 'New file name: '
        BYTE 0

PNTFPD  BYTE >0D,>0A
        TEXT 'Program description: '
        BYTE >0D,>0A
        BYTE 0

PNTNFPD BYTE >0D,>0A
        BYTE >0D,>0A
        TEXT 'New program description: '
        BYTE >0D,>0A
        BYTE 0

*The following elements need to reside in RAM, when the rest is written to EPROM.

        AORG >F000

CSBTRN1 BSS 2           Return address stored from R11 before BL'ing other routines.
CSBTRN2 BSS 2           Return address stored from R11 before BL'ing other routines.

IDEPM01 BSS 2           Number of sectors to read or write.
IDEPM02 BSS 2           Sector number to read or write.
IDEPM03 BSS 2           IDE command to perform.
IDEPM04 BSS 2           Start address of buffer to read into or write from.
IDEPM05 BSS 2           End address of buffer (calculated in this routine).
IDERTN  BSS 2           Return address stored from R11 before BL'ing other routines.

WSREG   BSS 32          Workspace registers.

CLMAX   EQU 70          Length of command line and text string entry buffer.
CLBUFF  BSS CLMAX       Command line and other text string entry buffer. Also used to hold some
*                       status values in other routines.
        EVEN

CBASSA  DATA 0          Save/load start address passed in by Cortex BASIC.
CBASEA  DATA 0          Save end address passed in by Cortex BASIC. Load end address passed back to BASIC.

SAVOVW  DATA 0          Flag as to whether a file already being saved already exists and is to be
*                       deleted when the new copy is saved.

IDEBUF1 BSS 512         First 512-byte sector buffer.
IDEBUF2 BSS 512         Second 512-byte sector buffer.
IDEBUF3 BSS 512         Third 512-byte sector buffer.

        END
```