GenRef
v1.00

MDOS Reference guide.

DSR Library

(C) Copyright 1989
J. Paul Charlton

DSR - CONTENTS

DSR OVERVIEW

The MDOS device drivers (Device Service Routines) are used to transfer data from your program to a mass storage device, from a mass storage device to your program; and to transfer data between your program and a character oriented device like a terminal, modem or printer. The device drivers are also used to control device characteristics such as baud rate, creating subdirectories, and initializing new media.

Available Devices

DSK1 ... DSK9	DSK1 through DSK9 are block oriented devices with a floppy disk directory structure. Each of these devices is limited to 3200 sectors. The assignment of these names to actual drives/ramdisks can vary depending on your hardware configuration.
HDS1 ... HDS3	HDS1 through HDS3 are block oriented devices with a winchester disk directory structure. See your HFDC owner's manual for more information on these devices.
SCS1 ... SCS3	SCS1 through SCS3 are block oriented devices with a winchester disk directory structure. See your SCSI owner's manual for more information on these devices.
RS232, RS232/1, RS232/2 RS232/3, RS232/4	RS232 through RS232/4 are serial bidirectional asynchronous character devices, both input and output is buffer with spoolers whose size you configured with your AUUTOEXEC file.
PIO, PIO/1, PIO/2	PIO through PIO/2 are parallel bidirectional character devices, usually used to send data to a printer. Both input and output is buffered with spoolers whose size was set with your AUTOEXEC file.
WDS1 ... WDS4	WDS1 through WDS4 are used to address the Personality Card winchester disk system. These devices do not support all MDOS DSR operations.

CALLING DSR FUNCTIONS

The MDOS device drivers must be called from within a machine code program running as a task under MDOS. You pass arguments to the DSR using a small area of memory known as a PAB (Peripheral Access Block.) The primary parts of a PAB are

an opcode, an error flag, and a full filename, other areas
of the PAB have meanings which are specific to the file
operation you wish to perform.

    The MDOS device drivers are invoked from a machine code
program when software trap number zero (XOP 0) is called
with a library number of 8.   The calling program's R0 must
contain the 16-bit address of the PAB at the time of the
XOP. The calling program's R0 must be located between >A000
and >FFD8 in memory, and should be located in the PAD RAM at
>F000 if possible.   The following code fragment will rename
a file on floppy drive #1.

```
          LI        R0,PABADR
          XOP       @EIGHT,0
          MOVB      @PABADR+2,R0     check for error
          JNE       ERROR
*
          BL        @PRINT
          TEXT      'File rename successful'
          BYTE      >0D,>0A,0
*
* ...
*
NEWNAM    TEXT      'NEWFILE'
PABADR    BYTE      >0D         rename opcode
          BYTE      >00
          DATA      0
          DATA      NEWNAM     assumes NEWNAM is mapped in
          DATA      0,0,0,0
          DATA      NAMLEN
NAME      TEXT      'DSK1.OLDFILE'
NAMLEN    EQU       $-NAME
*
```

    In the preceding example, three hidden assumptions were
made.   First, it is assumed that the program's registers are
correctly located, with the Workspace Pointer register of
the 9995 containing a value from >A000 to >FFD8.   Second, it
is assumed that PABADR is located on a page which is
currently mapped into a memory page which has the same 16-
bit address page number as its Virtual address page number
(read   the   section on Memory Management.)   The third
assumption is that NEWNAM is actually at the virtual address
NEWNAM, not in some overlay segment with a different virtual
address.

Always check these three assumptions in your own programs!!!

OPEN

Function   For block devices, such as hard disk and floppy
                            disk, this operation must be used to prepare a
                            structured file for access with READ and WRITE
                            opcodes.   You   must   close   the file before
                            terminating your task. No other task in MDOS is
                            permitted to open the file while your task has it
                            open.

                            For character devices, such as RS232 and PIO, this
                            operation is only used to change the operating
                            modes of the device port you specified as part of
                            the   file   name.   Multiple tasks are allowed
                            simultaneous access to the   RS232   and   PIO
                            ports...user beware.   You do not need to open a
                            character device before reading and writing to it.

Pab format   Parameters passed to OPEN

                            byte offset   size   parameter

                            0   1 byte   opcode = >00
                            1   1 byte   mode flags
                            6   2 bytes   records to reserve for newly
                                                       created file
                            8   2 bytes   record length
                            15   1 byte   name length
                            16   string   file name

                            Parameters returned from OPEN

                            byte offset   size   parameter

                            2   1 byte   error code
                            8   2 bytes   your record length
                            12   2 bytes   actual record length

Parameter description

Opcode   >00 is the opcode for the OPEN function in the DSR.

                            This opcode only applies to structured files with
                            the display or internal attributes. You will get
                            an error if you attempt to open a non-structured
                            file in this manner.

Flag byte

Bit   Meaning
0 (lsb)   0 = Sequential access   file,   you   must   use
                sequential access for files with variable
                record lengths.
           1 = relative access file, this can only be used
                with fixed record length files.

2,1   00 = Update mode, can only be used with fixed
                record length files. You are allowed to READ
                and WRITE records to the file.
           01 = Output mode, this is used to create a new
                file.   You are only allowed to WRITE records
                to this file. If the file already existed on
                disk, the old contents of the file will be
                forgotten. You will get an error if there is
                already a Protected file with the name you
                specified.
           10 = Input mode, this is used prepare for reading
                from an existing file. You will get an error
                if the file doesn't already exist.
           11 = Append mode, this is used to prepare for
                writing to the end of an existing file, or
                for creating a new file. You will get an
                error if you try to use this with a fixed
                record length file.

3   0 = Use display format data
           1 = Use internal format data

                This bit has no effect on the data actually
                stored in the file. It is provided by the
                programmer as an indication of the type of
                data in the file. You will get an error if
                this flag does not match the attributes of a
                file you are opening for update, input, or
                append modes.

4   0=   File will use variable record lengths
           1 = File will use fixed record lengths

                This flag will affect how data is stored in
                the file, and whether you can use relative
                record access within the file.

5,6   00   RESERVED, set to ZERO.

7   0   IGNORED
           1   For RS232 and PIO devices only, setting this
                bit to one will cause the driver to change
                the mode of the port as specified in the
                switches you placed after the device name in
                the filename. The mode will be set after all
                data currently in the output spooler for the
                specified device has been processed.

Error code

>00   No error occurred, the file is open and ready
                for reads and writes.
>20   Write Protection violation. A file you were
                trying to open for UPDATE, APPEND, or OUTPUT
                could not be created because the floppy disk
                has a write-protect tab. Alternatively, an
                existing file which you were trying to open
                for   UPDATE,   APPEND,   or OUTPUT has the
                "protected" attribute   bit   set   in   its
                directory entry.
>40   Invalid attributes. You specified a record
                length which was different than that of an
                existing file. Or, you tried to open a fixed
                file in APPEND mode. Or, the attributes you
                specified in the mode byte do not agree with
                the file attributes of an existing file on
                the disk.
>80   Out of space.   There are not enough free
                sectors on the device to create the file you
                specified;   or,   the   directory on   the
                specified device already has the maximum of
                127 entries; or, there are too many files
                open in MDOS, by your task and other tasks.
>C0   Media   error.   For   some   reason,   MDOS
                encountered   an   unrecoverable   error when
                trying to access the specified device.   For
                floppy disks, this could mean that the drive
                is empty. For hard disks, this means that
                there is no device present, or that MDOS was
                unable to read a needed sector from the hard
                drive.
>E0   General purpose error. An error which didn't
                fit any of the previous descriptions. You
                will get this error if a file you specified
                for INPUT mode does not yet exist. You will
                also get this error if any subdirectory
                specified as part of the filename does not
                exist on the specified device.

Reserved Records   If this parameter is set to zero during the
                          OPEN call, no space will be reserved for a
                          newly created file. You can reserve sectors
                          for a file when you open the file by setting
                          this parameter to the number of records you
                          expect to put into a newly created file.
                          Reserving space during the OPEN call can
                          greatly decrease the time needed to write
                          records to a new file since MDOS doesn't have
                          to spend time allocating more sectors for the
                          file each time you want to write more records
                          to the file. For files with record lengths
                          less than 255 characters, the number of
                          sectors reserved is calculated with the
                          following formula: Sectors =   Records/
                          INT(256/Record_Size).

For files with record lengths longer than 255 characters, the number of sectors reserved is calculated with the formula: Sectors = INT((Records*Record_Size)/256). Note that if you specified a record length of ZERO, a record length of 80 will be used by MDOS. For variable files with record lengths less than 255, the record size is the record length plus one. For fixed files, the record size is the same as the record length. For variable files with records lengths greater than 255, the record size is the record length plus two.

Record Length   This parameter specifies the default record length for a file. If you are creating the file, and this parameter has a zero value, MDOS will use a record length of 80 characters. If the file already exists, passing a zero value for this parameter will cause MDOS to use the record length of the existing file. If the file already exists, and you specify a non-zero value for this parameter which is different from the record length of the file, you will get an error. Specifying a zero value will always open any structured file.

On return, this will be the same as you set it, unless you initially specified a zero length. If you specified a zero length, and the file already existed, this will have the true record length of the file. If you specified a zero length, and you were creating a new file, this will have a value of 80.

Actual Length   On return, if the file you were trying to access already existed, this will contain the record length of the file. This is returned even if you get an error for specifying an incorrect record length for a file which already exists. If you just created the file, this will contain your record length.

Filename length   This is a count of the number of characters in the filename string.

Filename string   For block devices, such as disks and hard disks, this string must contain the name of the device you wish to access, followed by a list of the subdirectories separated by periods, followed by the name of the file you wish to access. (Example: "HDS1.SOURCE.UTIL.EXAMPLE") The length of the name, including the period separators,

Must be limited to 40 characters. For the OPEN call,
MDOS will not create directories as specified in
your pathname. If the specified directories did not
exist, you will get an error. Floppy disks are currently
limited to one subdirectory level. The individual names
for the subdirectories and your filename are limited to
ten characters, you will get an error if you try to use
more than ten characters.

For character devices, such as RS232 and PIO, this string
must contain the name of the device you wish to set for,
followed by a list of switches you wish to set separated
by periods. (Example: "RS232/2.BA=9600.DA=8.PA=N.CR")
Note that the switches are set only if you set bit
seven of the flag byte to a one. The switches will not
take affect until all output previously loaded into the
spooler for the specified device has been processed.

PIO   The following switch extensions may be used
with the PIO ports.

CR   Turn off carriage returns and line-feeds
            after each variable record sent.
LF   Turn off line-feeds after each variable
            record sent, each variable record is still
            followed by a carriage return.
NU   Print nulls after each variable record to
            allow for a low slew-rate printer.
IB   Reconfigure the spooler to recognize a
            printer with an Inverted-Busy handshake
            signal. (If your printer doesn't seem to
            work with MDOS, try turning this switch on.)
HS   Reconfigure the spooler to perform a full
            handshake with the printer for each byte sent
            (instead of just strobe.) (If your printer
            doesn't seem to work with MDOS, try turning
            this switch on.)

RS232   The following switch extensions may be used
with the RS232 ports.

CR   Turn off carriage returns and line-feeds
            after each variable record sent.
LF   Turn off line-feeds after each variable
            record sent, each variable record is still
            followed by a carriage return.
NU   Print nulls after each variable record to
            allow for a low slew-rate printer.
BA=baudrate   (110,300,600,1200,2400,4800,9600,19200)
DA=databits   (7,8)
PA=parity   (O,E,N) for (Odd,Even,None), respectively.
TW   Use two stopbits on transmission instead of
        one stopbit. (If your device doesn't seem to
        work with MDOS, try turning this switch on.)
CH   Check parity on each input character.

CLOSE

FUNCTION   This operation must be performed to close an
                                 open file.   It will cause MDOS to write any
                                 modified file buffers associated with the
                                 file; and if the file was modified, MDOS will
                                 write out a new copy of the file's directory
                                 entry.

                                 You must perform this operation on every file
                                 you have open before causing your program to
                                 terminate.

                                 This operation has no effect on a character
                                 devices such as RS232 and PIO.

PAB format   Parameters passed to CLOSE

                                 byte offset   size   parameter

                                 0   1 byte   opcode = >01
                                 15   1 byte   name length
                                 16   string   file name

                                 Parameters returned from CLOSE

                                 byte offset   size   parameter

                                 2   1 byte   error code

Parameter description

Opcode   >01 is the opcode for the CLOSE function in
                the DSR.

Error code

>00   No error occurred, the file is now closed, or
                the file wasn't even open.

>20   Write Protection violation. MDOS was unable
                to flush the file's buffers to disk. This
                should only happen if the user switched
                floppy disks while the file was open.

>C0   Media   error.   For   some   reason, MDOS
                encountered   an   unrecoverable   error   when
                trying to flush the buffers for the specified
                file. For floppy disks, this could mean that
                the user prematurely removed the disk from
                the drive. Alternatively, MDOS was unable to
                locate the place on the disk where the
                flushed data was to be written.

>E0   General purpose error. An error which didn't
                fit any of the previous descriptions.

READ

Function   This operation is used to transfer data from
                                the specified device/file to a buffer you
                                specify. For block devices such as floppy
                                and hard disk, the file must already be open.

                                You can read from a character device such as
                                RS232 at any time, no OPEN operation needs to
                                be performed.

                                Reading from a character device is designed
                                to accept input from a user typing at a
                                terminal. For variable record length files,
                                MDOS   will   interpret   certain   control
                                characters (described later in this section)
                                input from the device as editing commands for
                                the   current   input line, and display
                                appropriate changes on the user's terminal.

PAB format   Parameters passed to READ

                                byte offset   size   parameter

                                0   1 byte   opcode = >02
                                3   3 bytes   buffer address
                                6   2 bytes   record number
                                8   2 bytes   record length
                                10   1 byte   CPU/VDP flag
                                15   1 byte   name length
                                16   string   file name

                                Parameters returned from READ

                                byte offset   size   parameter

                                2   1 byte   error code
                                6   2 bytes   next record number
                                11   1 byte   ZERO
                                12   2 bytes   character count

Parameter description

        Opcode   >02 is the opcode for the READ function in
                    the DSR.

        Error code

        >00   No error occurred, the read was successful.
        >A0   You attempted to read past the end of the
                    file's previous contents. Your buffer does
                    not contain valid data.
        >C0   For block devices, media   error.   For some
                    reason, MDOS encountered an unrecoverable
                    error when trying read   data   from   the
                    specified file. For floppy disks, this could
                    mean that the user prematurely removed the
                    disk from the drive. Alternatively, MDOS was
                    unable to locate the place on the disk where
                    the data was to be read from.

                    For RS232 character devices, the hardware
                    detected an error such as incorrect parity,
                    byte too long, character buffer overflow (a
                    character was lost.)
        >E0   General purpose error. An error which didn't
                    fit any of the previous descriptions. You
                    will get this error if the file you specified
                    is not currently open.

    Buffer Address This is where you specify the address to
                    which data is to be transferred.

                    For transfers to VDP RAM, only the lowest 17
                    bits of the 3 bytes are significant.

                    For transfers to CPU RAM, only the lowest 21
                    bits of the 3 bytes are significant. For CPU
                    RAM transfers, the lowest 13 bits are an
                    offset into one of your task's memory pages,
                    and the other 8 bits specify which of your
                    task's pages the transfer starts on. This
                    address is not necessarily the same as the
                    16-bit CPU address your task will use to
                    examine the data, depending on how your task
                    has altered the map of its execution memory
                    pages (see the section on Memory Management.)

    Record Number The record number is only valid for Fixed
                    record files.   You will get an error if you
                    set the record number higher than the highest
                    record ever written to the file.   It is
                    possible to read a buffer full of garbage if
                    you specify a record number which has never
                    been written to.

On return, the record number has been
Incremented by one from the value you passed.
Note that no distinction is made between
Relative and Sequential access for Fixed files.
All access to fixed files is treated as
sequential unless you change the record
number within your program.

Record Length   For block devices, your program should not
                  alter this value.

                  For character devices, set this value to the
                  length of your input buffer.

CPU/VDP Flag   If this byte is zero, data will be
                  transferred to a buffer in the memory
                  belonging to your task, at the address
                  specified in the buffer address. If this
                  byte is non-zero, data will be transferred to
                  VDP RAM.

Character Count   On return, this contains the number of
                  characters placed into your input buffer.
                  For fixed files on block devices, this is
                  always the same as the record length for the
                  file. If you were reading fixed records from
                  a character device, this value may be smaller
                  than the record length if no input was
                  available.

NOTE: For variable record length files on block
                  devices, this count can actually be larger
                  than the specified record length, for
                  compatibility with existing applications like
                  TI-Extended Basic. (The notable example is
                  I/V 163 Merge format files, which can have
                  records longer than 163 characters.)

Filename length   This is a count of the number of characters
                  in the filename string.

Filename string   For block devices, such as disks and hard
                  disks, this string must contain the name of a
                  file which you currently have opened. For
                  character devices, this is simply the name of
                  the device, all switches after the name are
                  ignored.

EDIT Characters   Editing characters available on character
                  input devices.

                  ENTER   code >0D terminates input
                  ECHO   code >12 redisplays current input line
                  DEL   code >7F deletes last character in line

---
WRITE

Function   This operation is used to transfer data from
                         a buffer you specify to the specified
                         device/file. For block devices such as
                         floppy and hard disk, the file must already
                         be open.

                         You can write to a character device such as
                         RS232 or PIO at any time, no OPEN operation
                         needs to be performed.

                         Writing to a character device with fixed
                         record   lengths   causes   the   number   of
                         characters you specified to be written to the
                         device,   no end of line markers (except
                         possibly NULLS) are added to the   data.
                         Writing to a character device with variable
                         record   lengths   causes   the   number   of
                         characters you specified to be written to the
                         device, possibly followed by a   carriage
                         return and a linefeed, just a linefeed, or no
                         extra characters, depending on the switches
                         you set with an OPEN call. NULLS may be
                         added to your data if you turned NULLS on
                         with an OPEN call.

PAB format   Parameters passed to WRITE

                         byte offset   size   parameter

                         0   1 byte   opcode = >03
                         3   3 bytes   buffer address
                         6   2 bytes   record number
                         8   2 bytes   record length
                         10   1 byte   CPU/VDP flag
                         15   1 byte   name length
                         16   string   file name

                         Parameters returned from WRITE

                         byte offset   size   parameter

                         2   1 byte   error code
                         6   2 bytes   next record number

Parameter description

Opcode   >03 is the opcode for the WRITE function in
                the DSR.

Error code

>00   No error occurred, the write was successful.
>20   Write protection. MDOS was unable to write
                data to the file. If the file was opened in
                OUTPUT mode, this means that the user has
                switched disks since the file was opened, and
                that the new disk is write-protected. For a
                file opened in APPEND or UPDATE mode, this
                probably means that the disk was write-
                protected from the outset.
>80   Out of space. The disk is full.
>C0   For block devices, media error. For some
                reason, MDOS encountered an unrecoverable
                error when trying read the sector usage
                bitmap information or trying to write data
                into the file itself. For floppy disks, this
                could mean that the user prematurely removed
                the disk from the drive. Alternatively, MDOS
                was unable to locate the place on the disk
                where the data was to be written.
>E0   General purpose error. An error which didn't
                fit any of the previous descriptions. You
                will get this error if the file you specified
                is not currently open.

Buffer Address This is where you specify the address from
                which data is to be transferred.

                For transfers from VDP RAM, only the lowest
                17 bits of the 3 bytes are significant.

                For transfers from CPU RAM, only the lowest
                21 bits of the 3 bytes are significant. For
                CPU RAM transfers, the lowest 13 bits are an
                offset into one of your task's memory pages,
                and the other 8 bits specify which of your
                task's pages the transfer starts on. This
                address is not necessarily the same as the
                16-bit CPU address your task will use to
                initialize the data, depending on how your
                task has altered the map of its execution
                memory pages (see the section on Memory
                Management.)

Record Number   The record number is only valid for Fixed
                record files. Writing to a record past the
                end of the current file contents will cause
                the file to be expanded, possibly causing a
                disk full error.

On return, the record number has been
incremented by one from the value you passed.
Note that no distinction is made between
Relative and Sequential access for Fixed
files. All access to fixed files is treated
as sequential unless you change the record
number within your program.

Record Length   For block devices, your program should not
                   alter this value.

                   For character devices, set this value to the
                   length of your output buffer.

CPU/VDP Flag   If this byte is zero, data will be
                   transferred from a buffer in the memory
                   belonging to your task, at the address
                   specified in the buffer address. If this
                   byte is non-zero, data will be transferred
                   from VDP RAM.

Filename length   This is a count of the number of characters
                   in the filename string.

Filename string   For block devices, such as disks and hard
                   disks, this string must contain the name of a
                   file which you currently have opened. For
                   character devices, this is simply the name of
                   the device, all switches after the name are
                   ignored.

RESTORE

Function   For block devices, such as hard disk and
                            floppy disk, this resets the file so the the
                            next record read will come from the beginning
                            of the file, or the next record written will
                            go at the beginning of the file. The only
                            exception is fixed record files with the
                            relative mode on, those files will be
                            restored to the record number you specify.

                            This is the only opcode which makes a distinction
                            Between sequential and relative access fixed files.

PAB format   Parameters passed to RESTORE.

```
byte offset   size        parameter
0             1 byte      opcode = >04
1             1 byte      mode flags
6             2 bytes     record number
15            1 byte      name length
16            string      file name
```

                            Parameters returned from RESTORE

```
byte offset   size        parameter
2             1 byte      error code
6             2 bytes     record number
```

Parameter description

        Opcode   >04 is the opcode for the RESTORE function

                        This opcode must be used on a file which was
                        opened with the OPEN opcode. It does not
                        apply to character devices such as RS232 and
                        PIO.

        Flag byte   Only the least significant bit has meaning to
                        the restore opcode, and only for fixed files.
                        If the least significant bit is set to one,
                        for a fixed record file, the file pointer
                        will be moved to the record you specified in
                        the record number. Otherwise, the file
                        pointer will be moved to the start of the
                        file.

Error code

>60   Bad opcode. You attempted to use the RESTORE
               operation on a character device.
>E0   General purpose error. An error which didn't
               fit any of the previous descriptions. You
               will get this error if the file you specified
               is not currently open.

Record number For fixed files with the relative access mode
               bit set, this is the record number for
               subsequent access to the file.   This is
               pretty useless, because you can always set
               the record number on any read or write.

               Upon   return,   this   contains   zero   for
               sequential access fixed files, and the record
               number you specified for relative access
               fixed files.

Filename length
               This is a count of the number of characters
               in the filename string.

Filename string
               For block devices, such as disks and hard
               disks, this string must contain the name of a
               file which you currently have opened.

LOAD

Function   For block devices, such as hard disk and
                           floppy disk, this operation is used to read a
                           program image file into memory at the buffer
                           address you specified.

                           For character devices, this operation is not
                           implemented, but there was some thought about
                           using the 1K XMODEM protocol to implement
                           this operation.

PAB format   Parameters passed to LOAD

                           byte offset   size   parameter
                           0   1 byte   opcode = >05
                           3   3 bytes   buffer address
                           10   1 byte   CPU/VDP flag
                           11   3 bytes   buffer size
                           15   1 byte   name length
                           16   string   file name

                           Parameters returned from LOAD

                           byte offset   size   parameter
                           2   1 byte   error code
                           6   1 byte   >00
                           7   3 bytes   image size

Parameter description

          Opcode   >05 is the opcode for the LOAD function in
                           the DSR.

                           This opcode can only be used to transfer an
                           entire program image from a block oriented
                           storage device to memory. You will get an
                           error if you try to load any other type of
                           file.
          Error code

          >40   Bad attributes. You attempted to load a file
                           which wasn't a program image.
          >60   Bad opcode. You attempted to load from a
                           character oriented device such as RS232 or
                           PIO.
          >80   Buffer overrun. The image file on disk is
                           larger than the maximum buffer size you
                           specified.
          >C0   Media   error.   For   some   reason,   MDOS
                           encountered an unrecoverable error while
                           trying to locate or read from the specified
                           file. This could mean that a floppy drive is
                           empty, or there is a bad sector on the floppy drive.

>E0   General purpose error. An error which didn't fit
                    any of the previous descriptions. You will get this
                    error if the file does not exist on the specified device.

Buffer Address   This is where you specify the address to which
                   data is to be transferred.

                   For transfers to VDP RAM, only the lowest 17 bits
                   of the 3 bytes are significant.

                   For transfers to CPU RAM, only the lowest 21 bits
                   of the 3 bytes are significant. For CPU RAM transfers,
                   the lowest 13 bits are an offset into one of your
                   task's memory pages, and the other 8 bits specify
                   which of your task's pages the transfer starts on.
                   This address is not necessarily the same as the
                   16-bit CPU address your task will use to examine the
                   data, depending on how your task has altered the map
                   of its execution memory pages (see the section on
                   Memory Management.)

Image Size   These three bytes give the actual size of the image
                   file on disk. This value is returned to you even if the
                   file was not loaded because it was larger than your
                   buffer. Note that image files loaded from the WDSx
                   personality card winchester are limited to 16384 -
                   (buffer_address MOD 8192) bytes in length.

CPU/VDP Flag   If this byte is zero, data will be transferred to a
                   buffer in the memory belonging to your task, at
                   the address specified in the buffer address. If this
                   byte is non-zero, data will be transferred to VDP RAM.

Buffer Size   You use these three bytes to inform MDOS of
                   the maximum number of characters you wish to
                   load as an image file. If the image file is
                   longer than the size you specify here, none
                   of the image file will be loaded, and you
                   will get an error.

Filename length   This is a count of the number of characters
                   in the filename string.

Filename string   For block devices, such as disks and hard
                   disks, this string must contain the name of
                   the device you wish to access, followed by a
                   list of the subdirectories separated by
                   periods, followed by the name of the file you
                   wish to access. (Example: "HDS1.SOURCE.UTIL.EXAMPLE").

                   The length of the name, including the period
                   separators, must be limited to 40 characters.
                   This file must already exist on the device
                   you are accessing.

SAVE

Function   For block devices, such as hard disk and
                                floppy disk, this operation is used to write
                                a program image file to disk from memory at
                                the buffer address you specified. If the
                                file doesn't already exist, a new file will
                                be created if there is room on the disk. If
                                the file already exists, it must be an
                                unprotected program image file, or you will
                                get an error. When you are saving an image
                                file, any subdirectories specified in the
                                filename must already exist, they will not be
                                created for you.   You will get an error if
                                the specified subdirectories do not already
                                exist.

                                For character devices, this operation is not
                                implemented, but there was some thought about
                                using the 1K XMODEM protocol to implement
                                this operation.

PAB format   Parameters passed to SAVE for image files.

                                byte offset   size   parameter
                                0   1 byte   opcode = >06
                                3   3 bytes   buffer address
                                10   1 byte   CPU/VDP flag
                                11   3 bytes   buffer size
                                15   1 byte   name length
                                16   string   file name

                                Parameters returned from SAVE.

                                byte offset   size   parameter
                                2   1 byte   error code

Parameter description

        Opcode   >06 is the opcode for the SAVE function in
                        the DSR.

                        This opcode can be used to transfer an entire
                        program image from memory to a block oriented
                        storage device.

Error code

>20   Write Protection violation. The file could
                not be created because the floppy disk has a
                write-protect tab. Alternatively,   the   file
                already   exists, and has the "protected"
                attribute set in its directory entry.
>40   Bad attributes. You attempted to save over
                an existing file which wasn't a program
                image.
>60   Bad opcode. You attempted to save an image
                on a character oriented device such as RS232
                or PIO.
>80   Out of space.   There are not enough free
                sectors on the device to create the file you
                specified;   or,   the   directory   on   the
                specified device already has the maximum of
                127 entries.
>C0   For block devices, media error.   For some
                reason, MDOS encountered an unrecoverable
                error when trying read the sector usage
                bitmap information or trying to write data
                into the file itself. For floppy disks, this
                could mean that the user prematurely removed
                the disk from the drive. Alternatively, MDOS
                was unable to locate the place on the disk
                where the data was to be written.
>E0   General purpose error. An error which didn't
                fit any of the previous descriptions. You
                will get this error if any subdirectory
                specified as part of the filename does not
                exist on the specified device.

Buffer Address This is where you specify the address   from
                which data is to be transferred.

                For transfers from VDP RAM, only the lowest
                17 bits of the 3 bytes are significant.

                For transfers from CPU RAM, only the lowest
                21 bits of the 3 bytes are significant. For
                CPU RAM transfers, the lowest 13 bits are an
                offset into one of your task's memory pages,
                and the other 8 bits specify which of your
                task's pages the transfer starts on. This
                address is not necessarily the same as the
                16-bit CPU address your task will use to
                initialize the data, depending on how your
                task has altered the map of its execution
                memory pages (see the section on Memory
                Management.)

CPU/VDP Flag   If this byte is zero, data will be
                 transferred from a buffer in the memory
                 belonging to your task, at the address
                 specified in the buffer address. If this
                 byte is non-zero, data will be transferred
                 from VDP RAM.

Buffer Size   You use these three bytes to inform MDOS of
                 the number of characters you wish to save as
                 an image file.

Filename length This is a count of the number of characters
                 in the filename string.

Filename string For block devices, such as disks and hard
                 disks, this string must contain the name of
                 the device you wish to access, followed by a
                 list of the subdirectories separated by
                 periods, followed by the name of the file you
                 wish to save into. (Example:
                 "HDS1.SOURCE.UTIL.EXAMPLE") The length of
                 the name, including the period separators,
                 must be limited to 40 characters.

CREATE DIRECTORY

Function   For block devices, this operation is used to
                              create a new subdirectory on the specified
                              device. You can create a new subdirectory by
                              using a filename which ends in a period
                              separator character. Any intermediate
                              subdirectories specified in your filename
                              must already exist, they will not be created
                              for you. A filename of
                              "HDS1.LEVEL1.LEVEL2.LEVEL3." would create a
                              subdirectory named "LEVEL3" only if
                              subdirectories "LEVEL1" and "LEVEL2" already
                              existed on "HDS1". Note that only one level
                              of subdirectories, containing three
                              subdirectories, is allowed on a floppy disk
                              device.

                              For character devices, this operation is not
                              implemented.

PAB format   Parameters passed to CREATE DIRECTORY

byte offset	size	parameter
0	1 byte	opcode = >06
15	1 byte	name length
16	string	file name

                              Parameters returned from CREATE DIRECTORY

byte offset	size	parameter
2	1 byte	error code
6	1 byte	>00

Parameter description

            Opcode   >06 is the opcode for the CREATE DIRECTORY
                              function in the DSR. This is the same opcode
                              as the SAVE function of the DSR, the DSR
                              distinguishes between the two uses by looking
                              for a period character at the end of the
                              filename you specified.

Error code

>20	Write Protection violation. The subdirectory could not be created because a floppy disk has a write-protect tab.
>60	Bad opcode. You attempted to create a subdirectory on a character oriented device such as RS232 or PIO.
>80	Out of space. There are not enough free sectors on the device to create the subdirectory you specified; or, the directory on the specified device already has the maximum of 114 subdirectory entries.
>C0	For block devices, media error. For some reason, MDOS encountered an unrecoverable error when trying read the sector usage bitmap information or trying to write data into the subdirectory itself. For floppy disks, this could mean that the user prematurely removed the disk from the drive. Alternatively, MDOS was unable to locate the place on the disk where the data was to be written.
>E0	General purpose error. An error which didn't fit any of the previous descriptions. You will get this error if any subdirectory specified as part of the filename (other than the last one in the filename, the one you wish to create) does not exist on the specified device.

Filename length   This is a count of the number of characters in the filename string.

Filename string   For block devices, such as disks and hard disks, this string must contain the name of the device you wish to access, followed by a list of the subdirectories separated by periods, followed by the name of the subdirectory you wish to create, followed by another period. (Example: "HDS1.SOURCE.UTIL.NEWDIR.") The length of the name, including the period separators, must be limited to 40 characters.

DELETE

Function   This operation serves three purposes.

                          For block devices, such as hard disk and
                          floppy disk, this can be used to delete a
                          file from a directory, returning sectors
                          allocated to the file into available space on
                          the disk.

                          On block devices, this operation can also be
                          used to remove a subdirectory from a disk if
                          the subdirectory is empty.

                          For character devices, this operation can be
                          used to flush the output spooler for the
                          device, causing all characters still in the
                          output spooler to be purged.

PAB format   Parameters passed to DELETE.

                          byte offset   size   parameter
                          0   1 byte   opcode = >07
                          15   1 byte   name length
                          16   string   file name

                          Parameters returned from DELETE

                          byte offset   size   parameter
                          2   1 byte   error code

Parameter description

          Opcode   >07 is the opcode for the DELETE function.

          Error code

          >20   Write Protection violation. For files, the
                          specified file could not be deleted because
                          the disk is physically write-protected, or
                          the "protected" mode bit is set in the file's
                          directory entry. For subdirectories, the
                          directory could not be deleted because the
                          disk is physically write protected, or the
                          directory still has subordinate files and
                          directories within it (the directory is not
                          empty.)

>C0   For block devices, media error. For some
                reason, MDOS encountered an unrecoverable
                error when trying to update the sector usage
                bitmap information or trying to write data
                into the subdirectory itself. For floppy
                disks, this could mean that the user
                prematurely removed the disk from the drive.
                Alternatively, MDOS was unable to locate the
                place on the disk where the data was to be
                written.

>E0   General purpose error. An error which didn't
                fit any of the previous descriptions. You
                will get this error if any subdirectory
                specified as part of the filename (other than
                the last one in the filename, the one you
                wish to create) does not exist on the
                specified device. You will also get this
                error if the specified file/subdirectory did
                not exist.

Filename length This is a count of the number of characters
                in the filename string.

Filename string To delete files on block devices, such as
                disks and hard disks, this string must
                contain the name of the device you wish to
                access, followed by a list of the
                subdirectories separated by periods, followed
                by the name of the file you wish to delete.
                (Example: "HDS1.SOURCE.UTIL.EXAMPLE") The
                length of the name, including the period
                separators, must be limited to 40 characters.

                To delete a subdirectory from a block device,
                such as disks and hard disks, this string
                must contain the name of the device you wish
                to access, followed by a list of the
                subdirectories separated by periods, followed
                by the name of the subdirectory you wish to
                create, followed by another period.
                (Example: "HDS1.SOURCE.UTIL.NEWDIR.") The
                length of the name, including the period
                separators, must be limited to 40 characters.

                To purge the output spooler on a character
                device such as RS232 or PIO, this string must
                contain the name of the device. Any switches
                and other characters following the device
                name will be ignored.

DELETE RECORD

Function   This operation is not currently implemented
                              in MDOS. It would be used to delete a record
                              from a key indexed file (possibly implemented
                              as a B-Tree.)

PAB format   Parameters passed to DELETE RECORD.

                              byte offset   size   parameter
                              0   1 byte   opcode = >08
                              15   1 byte   name length
                              16   string   file name

                              Parameters returned from DELETE RECORD

                              byte offset   size   parameter
                              2   1 byte   error code = >60

Parameter description

          Opcode   >08 is the opcode for the DELETE RECORD
                        function.

          Error code

          >60   Bad opcode. This opcode is not implemented
                        in MDOS.

          Filename length
                        This is a count of the number of characters
                        in the filename string.

          Filename string
                        This would be the name of a currently open
                        file on a block device.

STATUS

Function   For block devices, such as hard disk and
                            floppy disk, this operation gives you
                            information about the attributes of a
                            specified file, as well as information
                            regarding space available on the disk.

PAB format   Parameters passed to STATUS.

                            byte offset   size   parameter
                            0   1 byte   opcode = >09
                            15   1 byte   name length
                            16   string   file name

                            Parameters returned from STATUS

                            byte offset   size   parameter
                            2   1 byte   error code
                            14   1 byte   attribute byte

Parameter description

        Opcode   >09 is the opcode for the STATUS function.

                            This opcode can be used on any file, even if
                            it is open. It does not apply to character
                            devices such as RS232 and PIO.

        Error code

        >60   Bad opcode. You attempted to use the STATUS
                            operation on a character device.
        >C0   Media   error.   For   some   reason,   MDOS
                            encountered an unrecoverable error while
                            trying to locate or read from the specified
                            file. This could mean that a floppy drive is
                            empty, or there is a bad sector on the floppy
                            drive.
        >E0   General purpose error. An error which didn't
                            fit any of the previous descriptions.

Attribute byte

Bit   Meaning

0 (lsb)   0 =   Not at end of file.
           1=   At end of open file, read is not possible
                 without EOF error. Write is possible if the
                 file was opened for writing.

1   0=   There is room on the disk to expand the file.
           1=   Disk is full, there is no room to make the
                 file larger.

2   0=   If the file is a data file, it has fixed
                 length records.
           1=   If the file is a data file, is has variable
                 length records.

3   0=   If file exists, it is a data file.
           1=   If file exists, it is a program image file.

4   0=   If the file is a data file, it contains
                 display format data.
           1=   If the file is a data file, it contains
                 internal format data.

5   This bit is not used.

6   0=   If the file exists, it is protected against
                 write operations.
           1=   If the file exists, write operations are
                 allowed.

7 (msb)   0=   File exists, all other bits specify file
                 attributes.
           1=   File does not exist, all other bits must be
                 ignored.

Filename length
                 This is a count of the number of characters
                 in the filename string.

Filename string
                 For block devices, such as disks and hard
                 disks, this string must contain the name of a
                 file for which you want to obtain
                 information.

FILE ID

Function   The file ID can be transferred from a file to
                            your program via the BREAD function. You can
                            cause MDOS to create a new file with all of
                            the characteristics in a file ID provided by
                            your task by use of the BWRITE function.   The
                            file ID is transferred between MDOS and your
                            task is the buffer whose address you specify
                            in the "buffer address" field of the PAB used
                            by BREAD and BWRITE.

                            The file ID tells you and MDOS everything
                            there is to know about the file other than
                            where it is actually located on the disk.

File ID format in buffer

offset	size	
0	2 bytes	Extended record length for files with records longer than 255 bytes. All 16 bits are significant, so a record whose length is specified here can be from 256 to 65535 bytes long.
2	1 byte	File status flag bits.
(lsb)	0=	Data file.
	1=	Program image file.
1	0=	If data file, contains display format data.
	1=	If data file, contains internal format data.
2	RESERVED	
3	0=	File is not protected.
	1=	File is write protected.
4-6	RESERVED	
(msb)	0=	If data file, contains fixed length records.
	1=	If data file, contains variable length records.
3	1 byte	For data files, this is the number of records which can fit into on one sector. This byte is zero for program images files and files with record lengths longer than 256 bytes.
4	2 bytes	This is the 16 least significant bits of a 20 bit number representing the number of sectors reserved for the file. The four most significant bits are at offset 18 in the buffer.
6	1 byte	This is the number of bytes used in the last sector of the file. A zero value in this byte means that all 256 bytes in the last sector are used. This is used in determining the length of program image files, and to determine the EOF for variable record length files.

7   1 byte   This is the record length for data files with
                         records shorter than 256 bytes. This is zero
                         for program image files and data files with
                         records longer than 255 bytes.

8   2 bytes   These two bytes must be reversed to form a 16
                         bit number which is the least significant 16
                         bits of a 20 bit number used to determine the
                         EOF   mark   for   files.   The   four   most
                         significant bits are at offset 19 in the
                         buffer.

                         For files with fixed record lengths, this is
                         one greater than the highest record ever
                         written to in the file.

                         For program image files, this is the number
                         of sectors actually used in the file. This,
                         in conjunction with the number of bytes used
                         in the last sector, is used to determine the
                         number of bytes in the program image.

                         For variable record files, this is simply the
                         number of sectors actually used in the file,
                         and is only used when you want to APPEND data
                         to the file.

10   2 bytes   Date of file creation. This has three fields
                         packed into the 16 available bits. The 7
                         most significant bits are the last two digits
                         of the year. The next 4 bits are the month.
                         The 5 least significant bits are the day of
                         the month.   The date 12 May 1988 would be
                         encoded as $(88*512 + 5*32 + 12)$ = >B0AC

12   2 bytes   Time of file creation. This has three fields
                         packed into the 16 available bits. The 5
                         most significant bits are the hour, from 0 to
                         23.   The next 6 bits are the minute, from 0
                         to 59. The 5 least significant bits are the
                         seconds, divided by 2. The time 10:58:13
                         would be encoded as $(10*2048 + 58*32 + 13/2)$
                         = >5746

14   2 bytes   Date of file update. Same format as the
                         creation date.

16   2 bytes   Time of file update. Same format as the
                         creation time.

18   1 byte   This contains the most significant 4 bits of
                         the number of sectors reserved for the file.

19   1 byte   This contains the most significant 4 bits of
                         the number of sectors used by the file.

BREAD

Function   For block devices, such as hard disk and
                                    floppy disk, this operation is used to read
                                    sectors directly from the disk or from within
                                    any type of file located on the disk into
                                    memory at the buffer address you specified.
                                    It can also be used to read subdirectory
                                    headers from a floppy disk.

                                    With files on a block device, if you specify
                                    a sector count of zero, this can also be used
                                    to read the file's ID into the buffer you
                                    specified.

                                    The values returned in the PAB are defined to
                                    make error recovery routines easy to code.

PAB format   Parameters passed to BREAD.

Byte offset	size	parameter
0	1 byte	opcode = >0A
3	3 bytes	buffer address
6	2 bytes	LSW sector offset
10	1 byte	CPU/VDP flag
12	2 bytes	sector count
14	1 byte	MSB sector offset
15	1 byte	name length
16	string	file name

                                    Parameters returned from BREAD.

byte offset	size	parameter
2	1 byte	error code
3	3 bytes	updated buffer address
6	2 bytes	updated LSW sector offset
12	2 bytes	remaining sectors
14	1 byte	updated MSB sector offset

Parameter description

         Opcode   >0A is the opcode for the BREAD function in
                        the DSR.

Error code

>60   Bad opcode. You attempted to read from a
                character oriented device such as RS232 or PIO.
>A0   Read past end of file. When you are reading
                sectors directly from a disk, this error is
                reported when you attempted to read a sector
                number higher than any sector on the disk.
                When you are reading from a file, this error
                is reported to you when you attempt to read a
                sector past the last sector reserved for file
                (not at the last sector used by the file.)
>C0   Media error. For some reason, MDOS
                encountered an unrecoverable error while
                trying to locate or read from the specified
                file. This could mean that a floppy drive is
                empty, or there is a bad sector on the block
                device.
>E0   General purpose error. An error which didn't
                fit any of the previous descriptions. You
                will get this error if the file does not
                exist on the specified device.

Buffer Address This is where you specify the address to
                which data is to be transferred.

                For transfers to VDP RAM, only the lowest 17
                bits of the 3 bytes are significant.

                For transfers to CPU RAM, only the lowest 21
                bits of the 3 bytes are significant. For CPU
                RAM transfers, the lowest 13 bits are an
                offset into one of your task's memory pages,
                and the other 8 bits specify which of your
                task's pages the transfer starts on. This
                address is not necessarily the same as the
                16-bit CPU address your task will use to
                examine the data, depending on how your task
                has altered the map of its execution memory
                pages (see the section on Memory Management.)

                On return from the BREAD function with a
                non-zero sector count, the buffer address
                will be updated to point to the first byte in
                memory after the data you just loaded. If
                there were no errors this will be your
                initial buffer address + 256 * sectors. If
                there were errors, this will point to the
                where the first bad sector encountered would
                have been loaded.

Sector offset   The two bytes at offset 6 in the PAB, along
                with the byte at offset 14 in the PAB, form a
                20 bit sector offset within the specified
                file or device. Sector numbering starts at
                zero, not at one.

On return from the BREAD function, this is
updated to contain the offset of the sector
in   the   file   after   the   last   sector
successfully read.   If there was an error
while reading sectors, this will contain the
offset of the sector which MDOS was unable to read.

CPU/VDP Flag   If this byte is zero, data will be transferred
                   To a buffer in the memory belonging to your task,
                   at the address specified in the buffer address.
                   If this byte is non-zero, data will be transferred
                   to VDP RAM.

Sector count   To read sectors from a disk or from a file, this must
                   be set to a non-zero number which tells MDOS how many
                   sectors you want to read. To read a file ID from a
                   file into your buffer, you must set this to zero.

                   On return, this will contain the number of
                   sectors not read due to an error condition.

Filename length   This is a count of the number of characters
                   in the filename string.

Filename string   For access to a file on a block device, such
                   as disk or hard disk, this string must
                   contain the name of the device you wish to
                   access,   followed   by   a   list   of   the
                   subdirectories separated by periods, followed
                   by the name of the file you wish to access.
                   (Example: "HDS1.SOURCE.UTIL.EXAMPLE")   The
                   length of the name, including the period
                   separators, must be limited to 40 characters.
                   This file must already exist on the device
                   you are accessing. You must specify a sector
                   count of zero if you want to read the file ID
                   into your buffer.

                   For direct access to sectors on a block
                   device, this string must contain the name of
                   the device you wish to access, followed by a
                   period. (Example: "HDS1.")

                   For direct access to the pointers of a floppy
                   disk subdirectory, this string must contain
                   the name of the floppy disk device, followed
                   by a period, followed by the subdirectory
                   name, followed by another period. (Example:
                   "DSK1.SUBDIR.")   For   this   subdirectory,
                   reading from sector zero or sector one will
                   return information unique to that subdirectory
                   in the same format as sector zero and sector
                   one of the main floppy disk. Access to any
                   sector beyond sector one will simply read the
                   corresponding sector from the disk, as if no
                   subdirectory had been specified.

BWRITE

Function   For block devices, such as hard disk and
                                 floppy disk, this operation is used to write
                                 sectors directly to the disk or into any type
                                 of file located on the disk from memory at
                                 the buffer address you specified.

                                 With files on a block device, if you specify
                                 a sector count of zero, this can also be used
                                 to create a new file with the characteristics
                                 described in a file ID in the buffer you
                                 specified.

                                 The values returned in the PAB are defined to
                                 make error recovery routines easy to code.

PAB format   Parameters passed to BWRITE.

                                 byte offset   size   parameter
                                 0   1 byte   opcode = >0B
                                 3   3 bytes   buffer address
                                 6   2 bytes   LSW sector offset
                                 10   1 byte   CPU/VDP flag
                                 12   2 bytes   sector count
                                 14   1 byte   MSB sector offset
                                 15   1 byte   name length
                                 16   string   file name

                                 Parameters returned from BWRITE.

                                 byte offset   size   parameter
                                 2   1 byte   error code
                                 3   3 bytes   updated buffer address
                                 6   2 bytes   updated LSW sector offset
                                 12   2 bytes   remaining sectors
                                 14   1 byte   updated MSB sector offset

Parameter description

            Opcode   >0B is the opcode for the BWRITE function in
                         the DSR.

Error code

>60   Bad opcode. You attempted to write to a
               character oriented device such as RS232 or PIO.
>80   Disk full. When you were creating a new
               file, there weren't enough sectors left on
               the disk to create the file. Alternatively,
               the specified subdirectory already had 127
               file entries, and your new file could not be
               added to the directory.
>A0   Write past end of file. When you are writing
               sectors directly to a disk, this error is
               reported when you attempted to write to a
               sector number higher than any sector on the
               disk. When you are writing to a file, this
               error is reported to you when you attempt to
               write to a sector past the last sector
               reserved for file (not at the last sector
               used by the file.)
>C0   Media   error.   For   some   reason,   MDOS
               encountered   an   unrecoverable error while
               trying to locate the specified file, or while
               trying to write to the file, or when trying
               to find more sectors for the file.   This
               could mean that a floppy drive is empty, or
               there is a bad sector on the block device.
>E0   General purpose error. An error which didn't
               fit any of the previous descriptions. You
               will get this error if the file does not
               exist on the specified device (unless you
               were just creating the file.)

Buffer Address This is where you specify the address from
               which data is to be transferred.

               For transfers from VDP RAM, only the lowest
               17 bits of the 3 bytes are significant.

               For transfers from CPU RAM, only the lowest
               21 bits of the 3 bytes are significant. For
               CPU RAM transfers, the lowest 13 bits are an
               offset into one of your task's memory pages,
               and the other 8 bits specify which of your
               task's pages the transfer starts on. This
               address is not necessarily the same as the
               16-bit CPU address your task will use to
               initialize the data, depending on how your
               task has altered the map of its execution
               memory pages (see the section on Memory
               Management.)

               On return from the BWRITE function with a
               non-zero sector count, the buffer address
               will be updated to point to the first byte in
               memory after the data you just saved. If
               there were no errors this will be your initial

                    buffer address + 256 * sectors. If there
                    were errors, this will point to the first
                    sector of data in your buffer which was not
                    saved to the block device.

Sector offset   The two bytes at offset 6 in the PAB, along with
                    the byte at offset 14 in the PAB, form a 20 bit
                    sector offset within the specified file or device.
                    Sector numbering starts at zero, not at one.

                    On return from the BWRITE function, this is updated
                    to contain the offset of the sector in the file
                    after the last sector successfully written.
                    If there was an error while writing sectors, this
                    will contain the offset of the sector which MDOS
                    was unable to write data into.

CPU/VDP Flag   If this byte is zero, data will be transferred
                    from a buffer in the memory belonging to your
                    task, at the address specified in the buffer
                    address. If this byte is non-zero, data will
                    be transferred from VDP RAM.

Sector count   To write sectors to a disk or to a file, this
                    must be set to a non-zero number which tells
                    MDOS how many sectors you want to write.

                    To create a new file, with the file ID from
                    your buffer, you must set this to zero.

                    On return, this will contain the number of
                    sectors not written due to an error condition.

Filename length This is a count of the number of characters
                    in the filename string.

Filename string For access to a file on a block device, such as
                    disk or hard disk, this string must contain the
                    name of the device you wish to access, followed
                    by a list of the subdirectories separated by
                    periods, followed by the name of the file you
                    wish to access.(Example: "HDS1.SOURCE.UTIL.EXAMPLE")
                    The length of the name, including the period
                    separators, must be limited to 40 characters.
                    This file must already exist on the device
                    you are accessing (unless you are in the
                    process of creating the file with your own
                    file ID.) You must specify a sector count of
                    zero if you want to create a new file with
                    the ID information in your buffer.

                    For direct access to sectors on a block
                    device, this string must contain the name of
                    the device you wish to access, followed by a
                    period. (Example: "HDS1.")

PROTECT

Function   For block devices, such as hard disk and
                           floppy disk, this operation can be used to
                           remove or set write-protection for the
                           specified file.

PAB format   Parameters passed to PROTECT.

                           byte offset   size   parameter
                           0   1 byte   opcode = >0C
                           2   1 byte   protection flag.
                           15   1 byte   name length
                           16   string   file name

                           Parameters returned from PROTECT.

                           byte offset   size   parameter
                           2   1 byte   error code

Parameter description

          Opcode   >0C is the opcode for the PROTECT function.

                           This opcode should only be used on a file
                           which is not open.

          Protection flag If this byte is zero, the specified file will
                           be made into an unprotected file. If this
                           byte is non-zero, the file will have the
                           write-protect bit set in its directory entry.

          Error code

          >20   Write Protection violation. For files, the
                           protection of the specified file could not be
                           changed because the disk is physically
                           write-protected.
          >60   Bad opcode. You attempted to use the PROTECT
                           operation on a character device.
          >C0   Media error. For some reason, MDOS
                           encountered an unrecoverable error while
                           trying to locate or update the directory
                           entry for the specified file. This could
                           mean that a floppy drive is empty, or there
                           is a bad sector on the floppy drive.
          >E0   General purpose error. An error which didn't
                           fit any of the previous descriptions. You
                           will get this error if the file you specified
                           does not exist on the specified device.

RENAME

Function   For block devices, such as hard disk and
                            floppy disk, this operation can be used to
                            rename a file or a subdirectory on the
                            device. It can also be used to change the
                            volume label on a block device. After a
                            rename operation of a file or subdirectory,
                            the filename in the PAB will be updated to
                            reflect the new name of the file or
                            subdirectory.

PAB format   Parameters passed to RENAME.

```
byte offset   size        parameter
0             1 byte      opcode = >0D
3             3 bytes     buffer address
10            1 byte      CPU/VDP flag
15            1 byte      name length
16            string      file name
@buffer       10 bytes    new file name
```

                            Parameters returned from RENAME

```
byte offset   size        parameter
2             1 byte      error code
15            1 byte      name length
16            string      new file name
```

Parameter description

        Opcode   >0D is the opcode for the RENAME function.

                        This opcode should only be used on a file
                        which is not open.

Error code

>20   Write Protection violation. The name of the
                specified file or subdirectory could not be
                changed because the disk is physically
                write-protected.
>60   Bad opcode. You attempted to use the RENAME
                operation on a character device.
>C0   Media   error.   For   some   reason,   MDOS
                encountered an unrecoverable error while
                trying to locate or update the directory
                entry for the specified file. This could
                mean that a floppy drive is empty, or there
                is a bad sector on the floppy drive.
>E0   General purpose error. An error which didn't
                fit any of the previous descriptions. You
                will get this error if the file you specified
                does not exist on the specified device.

Buffer Address This is where you specify the address from
                which MDOS will obtain the new name for the
                file.

                For transfers from VDP RAM, only the lowest
                17 bits of the 3 bytes are significant.

                For transfers from CPU RAM, only the lowest
                21 bits of the 3 bytes are significant. For
                CPU RAM transfers, the lowest 13 bits are an
                offset into one of your task's memory pages,
                and the other 8 bits specify which of your
                task's pages the transfer starts on. This
                address is not necessarily the same as the
                16-bit CPU address your task will use to set
                the filename, depending on how your task has
                altered the map of its execution memory pages
                (see the section on Memory Management.)

New name   The buffer must contain ten characters for
                the new name of the file or directory. If
                the name is shorter than ten characters, you
                must add enough trailing spaces to the name
                to make the buffer contain ten characters.

CPU/VDP Flag   If this byte is zero, data will be
                transferred to a buffer in the memory
                belonging to your task, at the address
                specified in the buffer address. If this
                byte is non-zero, data will be transferred to
                VDP RAM.

Filename length This is a count of the number of characters
                in the filename string.

                On return from the RENAME operation, this
                contains the length of the new filename.

Filename string   To rename a file on a block device such as
                   disks   and hard disks, this string must
                   contain the name of the device you wish to
                   access,   followed   by   a   list   of   the
                   subdirectories separated by periods, followed
                   by the name of the file you wish to rename.
                   (Example: "HDS1.SOURCE.UTIL.FILE") The length
                   of the name, including the period separators,
                   must be limited to 40 characters.

                   To rename a subdirectory on a block device
                   such as disks and hard disks, this string
                   must contain the name of the device you wish
                   to   access,   followed by a list of the
                   subdirectories separated by periods, followed
                   by the name of the subdirectory you wish to
                   rename,   followed   by   another   period.
                   (Example: "HDS1.SOURCE.UTIL.DIRECT.")   The
                   length of the name, including the period
                   separators, must be limited to 40 characters.

                   To change the name of a volume in a block
                   device, this string must contain only the
                   name of the device, and the buffer must
                   contain the new name for the volume.

FORMAT

Function   For floppy disk block devices, this operation
                                is used to initialize blank diskettes, or to
                                reinitialize used diskettes (erasing the
                                previous contents of the disk.)

PAB format

                                Parameters passed to FORMAT.

                                byte offset   size   parameter
                                0   1 byte   opcode = >0E
                                3   1 byte   tracks
                                4   1 byte   skew
                                5   1 byte   interlace
                                6   1 byte   density
                                7   1 byte   sides
                                15   1 byte   name length
                                16   string   device name

                                Parameters returned from FORMAT

                                byte offset   size   parameter
                                2   1 byte   error code
                                8   2 bytes   sector count
                                11   1 byte   actual sides
                                12   1 byte   sectors per track

Parameter description

          Opcode   >0E is the opcode for the FORMAT function.

                              This opcode should only be used on a device
                              which does not have any open files.

Error code

>20	Write Protection violation. The name of the specified file or subdirectory could not be changed because the disk is physically write-protected.
>60	Bad opcode. You attempted to use the FORMAT operation on a character or winchester device.
>C0	Media error. For some reason, MDOS encountered an unrecoverable error while trying to initialize the disk. This could mean that a floppy drive is empty, there is a bad sector on the floppy drive, or the disk drive is defective.
>E0	General purpose error. An error which didn't fit any of the previous descriptions.

Tracks   This is the number of tracks per side to
                format on the specified device. For double
                sided disks, this must be either 40 or 80.
                For single sided disks, this can range from 1
                to 40, or 80. Track counts between 40 and 80
                can not be used. Track counts larger than 80
                cannot be used.

Skew   This is the rotational delay between the last
                sector on a specified track, and sector zero
                on the next track, specified in sectors.
                This delay can be optimized so that sector
                zero of the subsequent track is ready as soon
                as a head step operation between tracks is
                complete, eliminating an extra rotation of
                the floppy disk after a head step. One disk
                revolution is 200 milliseconds on a standard
                5.25" floppy disk. The optimum delay for a
                single-track   head   step   is   about   35
                milliseconds.   The optimum skew is generally
                (35/200) multiplied by the sectors per track,
                rounded up to the next highest integer.

                Skew is NOT the spacing between the zero
                sector on consecutive tracks.

Interlace   Interlace is used to adjust rotational delays
                between consecutively numbered sectors on the
                same track. After reading a sector from the
                disk, MDOS will immediately process data from
                the sector, returning information from the
                sector to the calling task. In the time MDOS
                spends processing the data, the disk is still
                turning and several sectors may pass the head
                of the disk drive. If one of those sectors
                which went by as MDOS was processing the data
                happened to be the next sector that MDOS
                needed, MDOS would have to wait for the disk to

spin around again before it could read that
sector from the disk. Interlace is used to
specify how many sectors are likely to go by
as MDOS is processing the current sector.
Interlace is the number of revolutions of the
disk it would take MDOS to read an entire
track, assuming that MDOS could read
consecutively numbered sectors from the track
without the extra rotational delay. With an
interlace of 2, consecutively numbered
sectors on a track would have another sector
between them, a sector which MDOS will miss
as it is processing the data from the first
sector.

Density   Setting this byte to >02 will cause MDOS to
               format the disk with 18 sectors per track on
               a double-density controller card.   Setting
               this byte to any other value will cause MDOS
               to format the disk with 9 sectors per track
               on all controller cards.

Sides   Setting this byte to >02 will cause MDOS to
               format both sides of a disk in a double-sided
               drive. If the drive is not double-sided, and
               you set this byte to >02, only one side of
               the disk will be formatted.   Setting this
               byte to any value other than >02 will cause
               MDOS to only format one side of the disk.

Sector Count   On return from the FORMAT operation, this is
               a 16 bit integer which indicates the number
               of sectors available on a freshly formatted
               diskette.

Actual sides   On return from the FORMAT operation, this is
               the number of sides of the disk that MDOS
               formatted. For a single sided drive, this
               will always be >01. For a double sided
               drive, the number here will depend on the
               number of sides you asked to be formatted.

Sectors   On return from the FORMAT operation, this
               byte will contain a >09 for single density
               disks, and >12 for double density disks.

Device name length
               This is a count of the number of characters
               in the filename string.

Device name   For block devices, such as disks and hard
               disks, this string must contain the name of
               the device you wish to format, followed by a
               period. (Example: "DSK1.")