1. INTRODUCTION

The cassette driver allows storage of data files on cassette tape. Two cassette recorders can be handled, with separate remote control of the motors on each, allowing reading from one and writing to the other. The files stored can contain any data, not just ASCII.

The files are stored on tape in "chunks" with each chunk being up to 4k bytes. Data is always written to or read from the tape in complete chunks, with the motor being stopped between chunks. However these chunks are buffered within the cassette driver so the user can read or write in any sized blocks or single characters.

Two data rates are provided for recording, these are approximately 1000 and 2400 baud. When a file is read in the speed is determined from the leader automatically.

The cassette driver makes use of the status line for displaying messages when it is loading and saving, and also to display the cassette loading meter. This loading meter can be used to set the level optimally when reading tapes.

2. CASSETTE FILE FORMAT

The details of how data is stored on cassette will be covered later. This section just describes the general format of a file on tape as it appears to the user.

A file consists of a "header chunk", followed by one or more "data chunks". Each chunk will be preceded by necessary information for syncronising the tape reading routine and establishing the speed, including a long leader tone. Details of this are given later.

2.1 The Header Chunk

The header chunk does not contain any data from the file. It contains the filename (which may be null), and a protection flag which can be used to prevent simple tape to tape copying (see later). The header chunk is used to identify the file.

2.2 Data Chunks

Each data chunk contains exactly 4k of data from the file, except for the last one which may have any amount from zero bytes up to 4k. This smaller chunk is used to mark the end of the file. The data within each chunk is split up into 256 byte blocks, with a CRC check done on each block. This ensures that a bad chunk will be rejected fairly quickly.

3. USING THE CASSETTE DRIVER

The cassette driver allows a maximum of two channels to be open to it at a time, one for reading and one for writing. A channel which is opened for reading cannot be written to and vice versa. A reading channel is opened by making an "open channel" call and a writing channel by making a "create channel" call. The cassette driver is thus the only built in device driver which distinguishes between these two EXOS calls.

3.1 Opening Channels for Reading

A channel can be opened for reading a file from tape by simply doing an "open channel" EXOS call with device name "TAPE:". The unit number controls the use of remotes (see below) and the filename is optional. The cassette channel will require a channel RAM buffer of 4k (enough for one data chunk) and an error (.NORAM) will be returned if there is insufficient RAM. An error will also be returned if a cassette read channel is already open (.2NDCH), or if there is a protection violation (.PROT - see below).

Assuming that all this was OK, the cassette driver will start the cassette motor (see section on remotes below), and start searching for a suitable header chunk. At this stage it will display the message "SEARCHING" on the status line.

When a header chunk is found, the name of this file from the header is examined. If it is not the same as the filename specified by the user, and if the user's filename was not null, then this is the wrong file. In this case the message "FOUND <filename>" will be displayed on the status line and the search for a suitable header chunk will continue.

If the filename is correct, or if the user's filename was null which means "load the first file found", then the message "LOADING <filename>" will be displayed on the status line and the "open channel" routine will return with a zero status code to indicate success, after stopping the cassette motor. Read character or read block function calls can now be made to read the data.

Unit number was ignored in version 2.0

At any point in this process, the STOP key can be pressed which will abort the searching and return immediately to the user. An internal flag will be set so that any attempt to read characters will result in an .EOF error. The user must close the channel.

The "LOADING" message is left on the status line until the channel is closed. It may of course be overwritten by the user.

3.2 Reading Data

Data can be read from a cassette read channel by simply making any combination of EXOS read character and read block function calls. Data from the tape is buffered in the 4k channel RAM area. When the channel is first opened this buffer will be marked as empty.

If there is data in the buffer when a read character call is made then the next byte will just be returned to the user immediately, and the buffer pointers adjusted.

If there is no data in the buffer when a read character call is made then another data chunk must be read from the tape. The tape motor will be started and the cassette driver will look for a chunk. When a chunk is found, if it is a header chunk then a .CCRC error is returned to the user and the end of file flag will be set so that no more bytes can be read. If a data chunk is found then the data from it will be read into the buffer with CRC checking.

Having read the chunk in successfully, the first character will be returned to the user. If this was the last chunk in the file then a flag is set which will prevent another chunk from being loaded. When this final chunk has all been read by the user, any further read character calls will result in .EOF errors being returned.

If a CRC error occurred in one of the 256 byte blocks in the chunk, then any previous blocks will be buffered as usual and can be read by the user. When all the valid blocks have been read, the next read character call will return a .CCRC error, and subsequent ones will return .EOF errors. No more data can be read from this channel.

The STOP key is tested all the time while data is being read from the tape. If it is pressed then it will cause an immediate return to the user, with the tape motor being stopped first. The end of file flag will be set so that any further read character calls will result in .EOF errors being returned. No data from the interrupted chunk, or any later chunks, can be read.

3.3 Creating Channels for Writing

A "create channel" EXOS call will be accepted as long as there is 4k of channel RAM available for the data buffer, a cassette write channel is not already open and there is no protection violation (see below). This is the same as for a reading channel.

Assuming that this is OK, the cassette driver will start the cassette motor and wait for a second or so for the tape speed to stabilise. The message "SAVING <filename>" will be displayed on the status line. After the delay the cassette driver will write out a header chunk for this new file, which will contain the filename, and then stop the motor. After this it will return to the user with a zero status code to indicate that the create was successful.

The STOP key is tested during the writing of the header chunk and if it is pressed then the write will stop immediately, the tape motor will be stopped and the channel will be marked as invalid so no data can be written to it. The channel will still be open and so must be closed by the user.

The "SAVING" message on the status line is left there until the channel is closed, although it may be overwritten by the user of course.

3.4 Writing Data

Data can be written to a cassette write channel by any combination of write character and write block calls. A write block call is treated exactly as if each character in the block was written individually. Data is written into a 4k buffer in channel RAM and is only written out to the tape when the buffer becomes full, or the channel is closed.

When a character is written, if is just added to the buffer and the buffer pointers adjusted. If there is still more room in the buffer then the cassette driver will return to the user immediately. If the buffer is now full then it will be written to tape as a data chunk.

The process of writing the data chunk is very similar to writing out the header chunk when the file was created. The motor is started and then there is a delay to allow it to come up to speed. The data chunk itself is then written out and the motor stopped. This process is interruptable with the STOP key.

3.5 Closing Channels

The cassette driver treats the "close channel" and "destroy channel" EXOS calls identically. When a write channel is closed then the final data chunk must be written out. This is done even if there are no bytes in the buffer, to mark the end of the file. If the STOP key was pressed while data was being written then the channel will have been marked as dead, and in this case the final block will not be written out.

For any cassette channel the status line will be blanked to remove the "LOADING" or "SAVING" message from the status line.

4. MISCELLANEOUS CASSETTE FEATURES

4.1 The STOP Key

The STOP key is tested whenever the cassette driver is actually accessing the tape, either for reading or for writing. Since the cassette driver disables normal EXOS interrupts while it is accessing the tape, it does not rely on the normal keyboard driver detection of the STOP key. Instead it tests for the stop key directly itself and simulates the keyboard's action. However it does not test the STOP_IRQ EXOS variable, so the STOP key will always halt cassette operations even if STOP_IRQ is non-zero.

When the STOP key is detected, a .STOP error will be returned to the user, and also a software interrupt will be caused with software interrupt code ?STOP. The channel which was being used will be marked as no longer valid so that the cassette driver will reject further read or write character calls.

4.2 Tape Output Speed and Level

The data rate for tapes being read is determined automatically from the leader signal, and the level has to be set by the user. However the speed and level for writing out of data are controlled by EXOS variables which must be set before opening the channel, unless the defaults are required

LV_TAPE determines the approximate peak to peak output level of the cassette driver as follows:

```
0 or 1  =>  20 mV
     2  =>  40 mV   (default)
     3  =>  80 mV
     4  => 170 mV
     5  => 350 mV
6...255 => 700 mV
```

SP_TAPE selects between two tape output speeds, a fast speed and a slow speed as follows: (The baud rates are approximate because the actual rate depends on the data since a one and a zero bit take different amounts of time.)

SP_TAPE=0 => Fast speed (approx. 2400 baud - default)
SP_TAPE<>0 => Slow speed (approx. 1000 baud)

4.3 Cassette Loading Level Meter

The cassette loading level meter is displayed whenever the cassette driver is searching for or reading a chunk from the tape. It derived directly from a hardware level detection circuit, separate from the cassette input circuit, and is displayed on the status line as either a red or a green block next to each other.

If the input level is increased it will become red and if the level is reduced it will become green. The optimum level is when it is just on the verge of changing between red and green, and it may in fact flash between the two as data comes in. Although this is the optimum level, the cassette input is not very sensitive to levels and a wide margin around this optimum level is acceptable.

The level meter is removed from the status line when a chunk has been read, to indicate that the tape is no longer being accessed.

The changing of the level meter is done by changing palette colours 2 and 3 of the status line. When the level meter is removed, these colours are restored to their original values. However in the meantime, if there is anything else on the status line it may change colour.

4.4 Remote Control Relays

The Enterprise is equipped with two remote control relays which enable it to control two tape recorders separately. The motor is started when the cassette driver wants to read or write a chunk and stopped as soon as this has been completed, or the STOP key pressed.

When a cassette channel is OPENed for reading or CREATEd for writing, the unit number decides which remote relay will be used for this channel. A unit number of zero or one will use remote-1 and any other value (notably two) will use remote-2. If the user gives no unit number then EXOS will set it to zero by default, so remote-1 will be used.

If two channels are opened then they will each use the remote defined by their unit number. It is generally useful to ensure that these are different so that each channel has exclusive use of one of the remotes, but there is no checking for this. The remote not being used by a cassette channel will not be disturbed at all.
Unit number was ignored in version 2.0

The remote relays can also be controlled by two EXOS variables (called REM1 and REM2), separately from the cassette driver. When one of these is changed then the appropriate relay will be set on or off as appropriate. The cassette driver always updates these variables when it uses the remotes so the variables always represent the current state of the relays. Like all other on/off EXOS variables, zero corresponds to "on" (motor going) and 0FFh corresponds to "off" (motor stopped).

The remote relays will always be set off when a reset I/O system occurs (see EXOS kernel specification). This occurs at a warm reset and when a new applications program takes control.

4.5 Use Without Remote Control

Cassette recorders without remote control can be used, provided the PAUSE button on the recorder is used at the correct times. For simply loading and saving programs, no pausing is necessary, assuming that the program doing the loading and saving is fast enough (IS-BASIC is).

For saving, pausing is only necessary to avoid long gaps between chunks, it is not essential. For loading, pausing is necessary to ensure that the data chunks are not missed while the machine is processing the data.

To help with this, when the cassette driver has finished reading a chunk, and there are more data chunks to follow, it displays a "PAUSE" message on the status line in place of the level meter. This message will remain until the next chunk is required, when it will be overwritten with the level meter again.

4.6 Cassette Sound Feedthrough

The EXOS variable TAPE_SND is used to control feedthrough of the tape input signal to the main sound output. If it is 0FFh then there is no feedthrough. If it is zero then the tape input signal will be fed to the hi-fi sound output and the internal speaker (if MUTE_SND is zero), but not to the headphone/cassette output (to avoid feedback problems). The default setting is on (zero).

4.7 Copying Protection

Since two cassette channels are supported, one for reading and one for writing, it is very easy to open appropriate channels and copy any file at all, regardless of its content, onto another tape. This can be done fairly easily with a BASIC "COPY" command. The cassette driver contains a facility for protecting a file against this very simple type of copying.

When a file is created, and the header written out, the current value of the EXOS variable PROTECT is copied into the header. If this is zero (the default) then the file is not protected. IF it is non-zero then the file is marked as a protected file.

When a read channel is opened, and the header is read in, the "protect" flag from the header is examined. If it is zero then no special action is taken. If it is non-zero then the cassette driver will not permit a write channel to be open at the same time as this read channel. Thus if a write channel is already open then this "open channel" call will be rejected with a .PROT error. If this "open channel" is accepted then further "create channel" calls will be rejected with the same error.

5. HARDWARE

The hardware will not be explained in any detail but the various ports and bit assignments are covered here.

The cassette data input, level detection input, remote control outputs and sound feed-through control are all available as bits on I/O ports as follows:

data input	port 0B6h	bit 7
level input	port 0B6h	bit 6
remote 1 output	port 0B5h	bit 6
remote 2 output	port 0B5h	bit 7
feedthrough toggle	port 0B5h	bit 5

The cassette output is in fact the same as the sound output, with the cassette out socket doubling as a headphone socket. The cassette output is therefore done by using the DAVE chip in D/A mode and so several of the DAVE chip registers are used. These will not be detailed here as they are defined in the DAVE chip specification.

6. CASSETTE DATA FORMAT

As mentioned before a file consists of a series of chunks, the first of which is a header chunk and the rest of which are data chunks. The section describes the format of a chunk in some detail. A header chunk is in fact a special case of a data chunk with a special block count as will be explained later.

6.1 Cassette Signals

Each byte is stored on tape as a series of 8 bits, least significant bit first, with no start or stop bits. Each bit is stored as a single cycle, with a different frequency to indicate whether the bit is set or clear. These frequencies are in a ratio of 2:3 which is large enough to allow the software to distinguish them when reading in, but small enough to keep the data rate high.

An intermediate frequency is used for the leader tone which comes before the data, and this leader is used to determine the data rate when reading.

A single cycle of lower frequency is used to indicate the start of the data and also establish the phase of the signal (since it may or may not be inverted).

When the data is being read back, all timing is done in terms of whole cycles rather than half cycles. This ensures that it is relatively insensitive to changes in duty cycle which can result from level changes (drop-outs etc.).

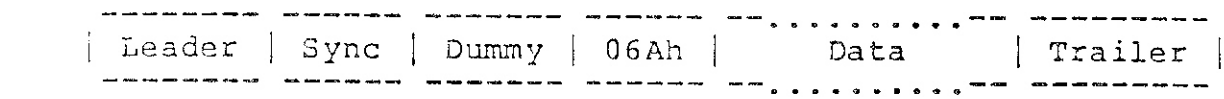
The actual whole cycle times and frequencies used for the two tape speeds are:

	Fast Speed	Slow Speed
Leader cycle	424us (2358 Hz)	1000us (1000 Hz)
One bit	344us (2907 Hz)	800us (1250 Hz)
Zero bit	504us (1984 Hz)	1200us (833 Hz)
Sync bit	696us (1437 Hz)	1600us (625 Hz)

6.2 Overall Chunk Format

Each chunk starts with a synchronisation sequence. This consists of several secords of leader frequency to allow the cassette recorder amplifiers and automatic recording level circuits to stabilise, and to establish the data rate. This is followed by a single low frequency sync cycle to establish the phase of the signal, and then one unused byte to recover from this one long pulse. The next byte always has the value 06Ah and is to ensure that false synchronisation does not occur.

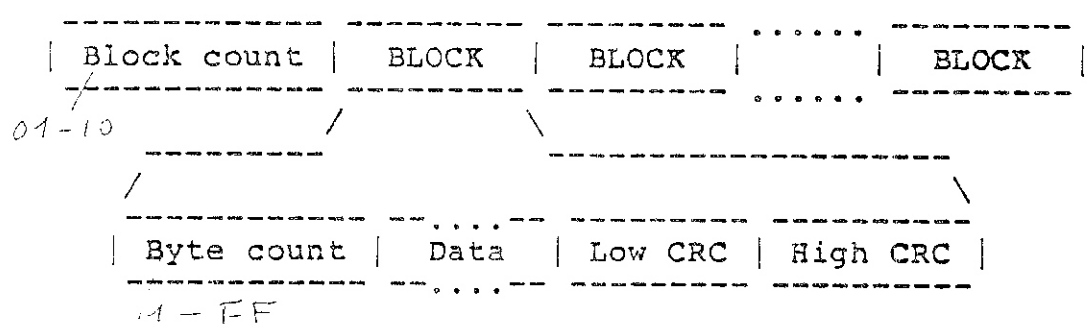
After this byte is the data of the chunk, followed by a few cycles of leader frequency (the trailer) to ensure a clean end to the data. The overall format is shown in this diagram:



6.3 Internal Chunk Format

The data within a chunk is split up into a maximum of sixteen 256 byte blocks, each starting with a byte count and ending with a two byte CRC check. These blocks are preceded in the chunk by a single, one byte, block count which defines how many blocks there are in the chunk.

The format of the data within a chunk is therefore as shown in this diagram:



A data chunk has a block count of 0 to 16, and each block is always 256 bytes long, except for the last block in the last chunk of the file. A block containing 256 bytes of data has a byte count of zero, which cannot be misinterpreted because zero is not a valid quantity.

6.4 Header Chunk Format

A header chunk always contains exactly one data block (of varying size), but has a block count of 255. This block count is used to recognise that it is a header chunk. The format of the data within the header chunk is:

```
byte 0:    Protection byte  (000h => protected file
                            0FFh => unprotecter file)
byte 1:  \
  .  .    \   Filename with length byte first.  Length
  .  .     >  byte is in range 0...28.
  .  .    /
byte N+1: /
```

6.5 CRC Checking

Each data block ends with a 16 bit CRC check. This is calculated by treating all bytes of the data block (not including the byte-count byte) as a bit stream. A 16 bit CRC register is initialised to zero at the start of the block and the following process carried out on each bit:

1) XOR the new bit into bl5 of the CRC register
2) If the new bl5 is set then XOR the CRC with 0810h
3) Rotate the CRC one bit left, putting bl5 into b0

Note that this is the same CRC algorithm as that used by the network driver.

7. QUICK REFERENCE SUMMARY

7.1 EXOS Calls

OPEN CHANNEL - Opens a read channel and gets the header chunk. Only one read channel allowed. Device name "TAPE:", Filename compared with file on tape (unless null). No EXOS variables need be set up before open.

CREATE CHANNEL - Opens a write channel and writes the header chunk. Only one write channel allowed. Device name "TAPE:", Filename written into header. EXOS variables LV_TAPE, SP_TAPE and PROTECT must be set up before create.

CLOSE/DESTROY CHANNEL - Treated identically. For a write channel will write out any buffered data.

READ CHARACTER/BLOCK - Only allowed for read channel. Read characters from buffer until empty, then read another data chunk from tape.

WRITE CHARACTER/BLOCK - Only allowed for write channel. Writes characters into buffer and writes it out to tape when it gets full.

READ STATUS - Returns C=0FFh at end of file or after an error.
C=0 otherwise.

SET STATUS - Not supported.

SPECIAL FUNCTION - No special functions

7.2 EXOS Variables

LV_TAPE - Tape output level (1...6)
SP_TAPE - Tape output speed. 0=fast, 0FFh=slow
PROTECT - Non-zero to write out protected file

TAPE_SND - Non-zero to suppress tape sound feed-through
REM1 - \ Control tape remote relays.
REM2 - / 0 for ON, 0FFh for OFF.

++++++++++ END OF DOCUMENT ++++++++++