Versalent

www.versalent.biz

RS232 INTERFACE TO MASS STORAGE

Mstor Manual

Preliminary -- Version 1.0x
Revised June 21, 2022

General Description :

Mstor is a small 3" X 2" X 1" module that allows a standard USB drive of up to 16 gigabytes to be accessed with non-USB systems through an RS232 port.. As a USB host it provides the FAT file system management allowing even the smallest of microcontrollers to use mass storage. RS232 data transfer speeds are slow compared to a standard USB connection, but systems which had no disk interface now have a means of accessing large storage. Mstor is intended for use as local or remote storage in any system with no USB accessibility.

Mstor communicates with a host through an RS232 interface using simple two-character commands. The initial commands to Mstor are always sent using no protocol, just a string of characters. And to accommodate hosts with small receive buffers, multiple data transfer protocols are provided. Data can be transferred in three ways:

1) a ***short sequence of bytes*** – no data grouping or data blocking is used, just a (short) sequence of bytes is sent without any protocol. This is used for commands to Mstor and for responses to the host when that response is just a few characters. Any commands which return more than 15 characters to the host use one of the following request/response protocol methods so that even hosts with tiny receive buffers can throttle their incoming data to prevent data loss. This communication scheme is referred to as 'no protocol'.

2) a ***grouped stream of bytes*** without any framing or error checking. This can be a large amount of data (such as an entire file) so it is transferred using a request/response protocol. A pre-set *group-size* defines how many bytes are returned each time the receiver sends a single character group-request to the sender. This scheme provides inherent 'data throttling' and is referred to as 'stream protocol'. Group-size can be set from 16 – 250 characters. The receiver cannot request the re-transmission of a group.

3) a ***block-framed format*** containing check characters for improved data integrity. Block sizes can be set from 16 to 250 bytes and the contained block check characters can be a 16-bit checksum or an optional 16-bit CRC. The receiver requests each block with a single block-request character, and can then ask for the next block, or a retry of the last block depending on its validation of the check characters. This scheme provides for 'data throttling' and error correction, and is referred to as 'block protocol'.

The FAT formatted USB drive is compatible with PC drives so data files can be accessed/edit/transfer between a PC and your non-USB system. Mstor can traverse through a directory structure using the 'C'hange Directory command and operate on files throughout the directory tree.

File and Directory names MUST be in 8.3 format – although Windows allows the use of long-winded names, Mstor does not. Directory paths such as SubDir1/SubDir2/filename.ext are limited to 60 characters.

The term 'characters' often refers to ASCII characters which are 7-bits – but to Mstor all characters are 8-bits so throughout this manual, the terms 'bytes' and 'characters' mean the same thing… 8 bits. ASCII non-printable characters such as ACK (0x06), NAK (0x15), CR (0x0d) are shown in this manual enclosed in { } brackets such as {ACK}{NAK}{CR}.

!DATA SAFETY!

As with PC's, random unplugging of an active USB drive can result in partial or complete loss of data and/or corrupted drive formatting. BEFORE REMOVING A USB DRIVE FROM Mstor, it should be powered-off using the &PF command. Visually check that the USB drive's LED goes out, and Mstor's power LED flashes quickly – indications that the drive is powered down. (Similar to Windows 'Safety Remove' function.)
Unplugging Mstor from its power source while its USB drive is active can also cause data loss. The USB drive should be powered-off first.

Detailed Description:

Commands and Modes:

All command identifiers are two characters .. an ampersand '&' followed by a command letter. That is followed by optional command parameters and terminated with a carriage return {CR}*. The response is {ACK}* for a valid and accepted command, or {NAK}E-xx for a rejected command. (xx is a two-decimal-digit error code). And for commands that return data, another {ACK} is sent after the completion of the data, or {NAK}E-xx on user-abort.
For disk read /seek/write/append operations, the host uses streaming mode commands (&R, &S, &W, &A) or block mode commands (&r, &s, &w, &a) . As above, Mstor first responds with {ACK} or {NAK}E-xx. Following an {ACK} the transfer begins and when the transfer is complete another {ACK} is returned. Stream and block timeouts are in effect to resume operation from any unexpected conditions.

*{ACK}=0x06 , {NAK} =0x15 are ASCII characters noting command acceptance/rejection. , {CR}=0x0D is the command terminator character

Mstor Modes of Operation:

Mstor is always in one of several modes:

1) **Command Mode** -- The Mstor command processor is idle, and it can receive and execute any of its available '&' commands from the table below.

2) **Write/Append Stream Mode** – Mstor has received a streaming Write or Append command (&W, &A) and is in the process of streaming all incoming data from the host to the USB drive. It will not accept any new & commands until this process completes or times out. The host initiates this mode, sends all data, then ends it by sending the predefined END_OF_SEQUENCE (EOS) string (not stored to disk). EOS is a message appended to the data stream to inform Mstor that there is no more data and the write process is complete.
   A fail-safe scheme for ending this mode uses the ' stream-timeout' which is set by the user. If this period of time passes with no characters received, Mstor automatically ends Stream Mode and re-enters Command Mode. It is fail-safe because it does not rely on the host, or receipt of a valid EOS. The Stream-Timeout is set in 100ms increments from 100ms to 25.4 seconds (1 to 254), or to 255 which sets the timeout period to 'indefinite' disabling the feature. (Refer to the 'o' command).

   After initiating the command and receiving an {ACK}, the host monitors for group-request characters.. Each time it receives any character (other than {NAK} = abort),

it sends another group of characters using the preset Group-Size (see g-command). *Groupsize* can range from 16 to 250 characters.

3) **Write/Append Block Mode** -- Mstor has received a Write/Append command (&w, &a) command and is expecting the host to send formatted blocks of data in response to 'N' (next block request) characters. Mstor can also send {NAK} to abort the mode, or any other character to request a block retry. After issuing the block request Mstor waits for up to the *blocktimeout* period for a block character to be received. The 2nd character received is the block's data-payload length so when [data-payload + 5] characters have been received the block is complete and Mstor validates/processes it. If the check characters are validated, the data is written to disk and the next block is requested.
If a block fails to validate, Mstor flushes the block from its buffer and requests a retry of the last block. An incomplete block (or no response) causes a block timeout -- anything received is flushed and a retry is requested. ***(The host must NOT send any un-solicited blocks/characters during this process.).*** When the host has no more data to send, or it wants to abort for any reason, it sends an empty block containing zero data bytes . Mstor responds with an {ACK} indicating that the command is complete and returns to Command Mode. Sample code available at www.versalent.biz/downloads/Mstorcode.c

4) **Read/Seek Stream Mode** -- Mstor has received a streaming Read or Seek command (&R, &S) and is in the process of streaming data from the specified file to the host It will not accept any new & commands until this process completes or is aborted. Mstor responds with {ACK}< filesize> (*filesize* is a string of decimal digits representing the total number of bytes in the file, or from the 'Seek' position, that will be returned.) Mstor will then stream a single group of bytes in response to any non-{NAK} single-character group-request from the host. {NAK} issued as a group-request aborts the process. There are no framing or error checking characters in the stream. *Groupsize* is pre-defined as 16-250 bytes per request which allows the host time to process each group from its buffer before requesting the next group. This request/group-response process continues until <filesize> bytes have been received, or the host sends a {NAK} to abort, or the Stream Timeout period passes with no host requests received. Mstor issues an {ACK}/{NAK}E-xx and returns to Command Mode.

5) **Read/Seek Block Mode** – Mstor has received a block Read/Seek command (&r, &s) and is in the process of returning formatted data blocks in response to host block-requests. When the command is issued with a valid filename, Mstor responds with an {ACK}, and awaits the first 'N' to return the next block. If the specified file, or directory could not be found, Mstor returns {NAK}E-xx (xx is a two decimal-digit error code). If a received block fails to validate, the host should request a retry (any character other than an 'N' or {NAK}). The host must implement its own timeouts as necessary to request a retry if an incomplete block is received after a period of time. (The 2nd byte of each block specifies the number of

data bytes contained). Mstor signals the end of data by sending a zero-byte block. Sample code available at www.versalent.biz/downloads/Mstorcode.c

6) **List Files Stream Mode** -- Mstor has received a stream List Files command (&Lpath/mask). After responding with an {ACK} it streams a list of file 'items' (see below) to the host. In this mode Mstor responds to {NAK} by aborting the process, or any other character to return the next group of characters. This request/group-response process continues until the last file item is sent, or the host aborts, or the Stream Timeout period passes without a host request. Immediately after sending the last file item, Mstor sends an {ACK} and returns to Command Mode.
If a path is included a file mask must also be included such as &L/mydir/nextdir/*.* . The list contains all the matching files items in the specified directory and each item contains:

Filename [SP] Create-date [SP] Create-time [SP] Size {CR}{LF}

Filename is 8.3 format
[SP] is an ASCII space
Create-date is MM/DD/YYYY format
Create-time is H:MM:AM format
Size is the number-of-bytes, or if > 999, number-of- kbytes i.e. 23k
{CR}{LF} newline separator to support display on a screen

7) **List Directories Stream Mode** – Mstor has received a stream Directory List command (&D). After responding with an {ACK} it streams a list of sub-directories items at this level in the tree. The list contains the same item format as above for file items format above and each group of characters arrives in response to the host issuing a group-request character. {NAK} aborts the process. After the last directory item Mstor sends an {ACK} and returns to Command Mode.

Streaming Mode Transfers:

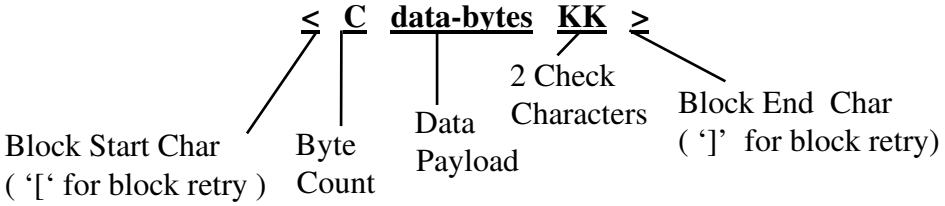
In streaming modes, just like with block-transfers, there are regular handshake exchanges between sender and receiver so the stream is not continuous as that could overrun a receiving buffer especially at high baud rates. When the receiver is ready for the next *groupsize* group of bytes, it sends a single character to the sender as a group-request. There is no error checking, or re-transmission capability. The sender delivers the next *groupsize* sequence of bytes or less if no data remains. Stream-Timeout provides built in lockup-prevention so the sender does not wait indefinitely for a request. Stream modes are the simplest to use but should only be used with data that can tolerate an occasional error, or files that contain their own CRCs. G*roupsize* should be set to a value less than or equal to

the size of the host's receive buffer and it should empty its buffer prior to sending an Mstor stream command to insure that there is room to buffer the response.

Block Mode Transfers:

A *block* refers to a framed payload of data bytes and each block adds 5 framing characters to that data. In block modes, the receiver requests each block – similar to streaming modes – however the receiver has the option to request a block-retry if a block's check bytes do not validate. Block requests also use a single character -- 'N' for next block, {NAK} to abort the command, or *any other character* to retry the last block. Blocks are framed with < > characters if in response to a 'N'ext block request, or framed with [ ] characters if it is the response to a retry. This scheme allows the receiver to correctly manage blocks even if the request characters themselves (which are not error-checked), get corrupted.

Block Format:



If the block is a 'retry' the framing characters are changed from < > to [ ] . See above.

C is a single binary byte noting the count of *data* bytes in the block. It can be no larger than Mstor's currently configured [blocksize -5] . Refer &B command.

Data-bytes .. the 8-bit binary data payload of this block.

KK represents two binary 8-bit check characters that form a 16-bit checksum, or a 16-bit CRC (if CRC enabled). It is computed from all block characters except the two check characters. The high byte is left-most.

< > or [ ] are the block framing characters. Most blocks use **< >** however if the receiver fails to validate a block's check characters, the receiver should request a block retry (sending any character except 'N'=next or {NAK}=abort). This system allows the receiver to handle even corrupted request characters -- by checking the framing characters which are included in the check-character calculations, the receiver can determine if its request was received correctly – and whether it has received a 'next' block, or a 'retry' block, and handle the data accordingly.

Block characters are transmitted starting at the left-most character.

Mstor Transfer Timeouts:

Transfer timeouts provide for recovery from unexpected states. Since Mstor operates asynchronously with its host, timeouts provide for automatic recovery from states that would otherwise lockup further interaction. For instance, if Mstor is waiting for data that never arrives, it recovers. There are two timeouts – one for streaming transfers, and one for block transfers. Each can be set from 100ms second to 25.4 seconds in 100ms increments (setting = 1 to 254), or disabled with a setting of 255.

Stream Timeout – In *Stream Write/Append Modes*: While Mstor is receiving a stream of data to write to disk, if the stream stops for more than this time period, Mstor terminates the mode, returns a {NAK}E-xx (xx = two decimal-digit error code), and returns to Command Mode. This mechanisms prevents Mstor from waiting indefinitely when no more data is going to arrive. The host should set this value to longer than the longest anticipated delay which can occur in its write-data-stream process.

In *Read/Seek Modes***:** While Mstor is returning a stream of disk data to the host, if the host stops making group-requests for longer than this period, Mstor terminates the mode, returns a {NAK}E-xx and returns to Command Mode.

Block Timeout -- In *Block Write/Append Modes*: While Mstor is requesting/receiving blocks to write to disk, if it makes a request and the host does not respond within this time period, Mstor re-issues its last request a maximum of 3 consecutive times before it issues a {NAK}E-xx error, aborts and returns to Command Mode.

In *Read/Seek Modes***:** While Mstor is reading the disk and responding to block requests from the host, if the host does not request a block within this time period, Mstor closes the file, issues a {NAK}E-xx error and returns to Command Mode.

Mstor End_Of_Sequence (EOS) string:

When **writing/appending data in streaming mode**s .. the data stream cannot inform Mstor when end-of-data occurs using any single character since they are all valid file characters. At the end of data the host could simply stop responding to group-requests and Mstor will timeout and terminate the mode – however this is an error-termination scheme not intended for normal operation. Instead, to inform Mstor when the data stream is complete and to exit gracefully without error, the host appends a predefined **End-Of-Sequence (EOS)** string to its data stream – a sequence of 16 characters that Mstor recognizes as the end of data. The **EOS** should be a sequence that is unlikely to

'accidently' appear in the data stream. Like a computer password, it should not contain sequential characters like ABC or 567…. A more random mix of 8-bit values is much less likely to exist in file data. Mstor continually scans the incoming data stream for this sequence. Additionally this sequence must be followed by at least 250ms of quiet-time (no characters received). When these conditions occur, Mstor closes the file and terminates the transfer mode. See the &E command below to use the default Mstor sequence, or create your own – this string/array is stored in non-volatile memory. If there is not 250ms of 'quiet' time after EOS is detected, Mstor continues to accept-and-write the incoming data.

[Note that streaming read commands do NOT use EOS because Mstor responses start with the number of bytes to be returned (the <filesize>). Refer to the &R command. Since the host knows how much data to expect, Mstor does not need to signal the end of data.]

Streaming vs Block Transfers:

Although the streaming read/write/append modes are the simplest to implement, they offer no error detection/correction. This makes these modes more suitable for data that is not 'bit-critical' – like audio streams, video data, text files or other data uses that can tolerate an occasional bit error. For data that must be accurate, the block modes are provided. Data is transferred in framed blocks that contain checksums/CRC's and allow the receiver to request retransmission of any block whose computed check characters do not match those within the block.

File System Limitations:

Mstor provides the filesystem which allows a host to read/write/edit/move/copy etc files throughout the USB drives directory structure. Wildcard masks are only allowed with the List Files (&L), and List Directories (&D) commands. This means that Mstor cannot operate on batches of files .. for instance cannot delete or copy groups of files like a PC. Filepaths can use either a foreslash '/' or backslash '\' character between directories. Mstor provides a real-time battery-backed clock to maintain file time stamps. The battery recharges during powered usage and lasts up to 3 months with power removed.

Mstor Data Transfer Speed:

Transfer speeds are limited by the serial port speed, disk seek/read speeds, and request/response interaction. At 9600 baud Mstor can stream 1k bytes per second and at 115k baud about 6k bytes per second. (streaming commands are &R, &W, &A). Block mode (&r, &w, &a modes) transfers are slightly slower since there are 5 overhead bytes surrounding the data in each block. The lowest baud rate of 1200. while very slow, does provides for the greatest distance to remote storage (120 bytes/sec at over 300 meters).

Directory and File Names:

All directory and filenames must be in 8.3 format. The top level (root) directory is identified by a single '/' or '\' character. Commands can be executed while in any directory, and can operate on files in any other directory using the appropriate path ahead of the filename. Paths starting with the '/' start at the root directory and follow the specified subdirectory path. So to delete Myfile which is two levels down under the root directory (while in any other directory), use :

&F/subdir1/subdir2/myfile{CR}

Or to delete Myfile which is two levels down below the current directory, use:

&Fsubdir1/subdir2/myFILE{CR}

Or to delete Myfile which is the current directory, no path is necessary, simply use:

&FMYFILE{CR}

Notice that the case of myfile in the above examples is different in every example. The file system is case-insensitive, so myfile matches with MYFILE and MyFile.

The double-dot syntax is available to specify a directory one level up from the current position in the directory tree. With Mstor currently in the botmdir directory:

(path from the root) /mydir/mydir1/nextdir.dir/botmdir

To move up one level issue .. &C..{CR} (moves to /mydir/mydir1/nextdir.dir)

To move up two more levels, .. &C….{CR} (moves to /mydir)

Path/filename strings can be a total of 50 characters long. However command strings are limited to 80 characters, so paths may have to be shortened for multi-file commands.

Directory Structure Complexity:

While there are no commands to return the entire directory tree, the &C{CR} command with no parameters provides the complete path from the root directory to the current directory. Ex: /mydir/mydir1/nextdir.dir/botmdir

If using a drive containing a complex, multi-branch directory structure, Mstor's host must have knowledge of that structure in order to navigate the tree. Without any visual representation of the directory tree, it is suggested that the tree structure be kept as simple as possible to keep navigation manageable.

Mstor Commands:

Commands consist of the ampersand '&' character followed by a command character, followed by command parameter(s), and ending with a {CR} carriage return. Commands must be issued one-at-a-time and the host must wait for a response of {ACK} or {NAK} which may be followed by additional information. Commands cannot be nested. That is, each must finish before a new command is presented. Every command results in a response even if it is just an acknowledge.
In the table of commands below, commands which change Mstor's mode are tinted. Commands consist of ASCII characters. Mstor generally responds with an ASCII {ACK} (0x06) or {NAK}(0x15) sometimes followed by additional information. Paths in disk-access commands are optional. {NAK}E-xx is a {NAK} error indicator followed by E-xx with xx being a two-decimal-digit error code.

As mentioned above, Mstor uses three protocols to communicate with a host. .. no protocol, stream protocol, and block protocol. These protocols allow for loss-less communication and each command lists which protocol is used. Again, no protocol is used in sending commands to Mstor so the protocol referenced applies after a command starts executing.

Mstor Command List

Command	Command Description
&W	**Write File Stream Mode .. uses stream protocol.** **Typical command: &Wpath/filename.ext{CR}** . Mstor responds with {ACK} for success or {NAK}E-xx If the specified filename exists in the specified directory, it will be overwritten. If it does not exist it will be created. Mstor then sends single character group-request characters and expects the host to respond with a sequence of *groupsize* characters which it writes to disk. When the END_OF_SEQUENCE (EOS) string is received. or a stream timeout occurs Mstor ends the mode. There is no error checking. The file remains open for data pauses of up to the *streamtimeou*t period (which can be indefinite). When Mstor receives EOS it responds with {ACK}. For stream timeouts or command aborts it responds with {NAK}E-xx . Mstor closes the file and returns to Command Mode.
&w	**Write File Block Mode… uses block protocol.** **Typical command: &wpath/filename.ext{CR}** . Mstor responds with {ACK} or {NAK}E-xx. If the specified filename exists it will be overwritten. If it does not exist it will be created. On {ACK} the host then awaits an 'N' requesting the first block. After each validated block Mstor sends 'N' to request the next block, or 'Y' to retry that block. This continues until the host sends a zero-length block. Block formatting: **< C data-chars KK >** … see above The < > characters are framing characters. For a retry the < > frame characters are replaced with [ ]. C is the one byte (binary) count of the data bytes in the block. KK is a high-byte-first character pair representing the continuous sum (16-bit checksum) or 16-bit CRC of all characters *except* the check characters.. There are C data characters contained in the block and C has a maximum value set by *blocksize*.. While in this mode, no other files can be opened, and no other commands can be initiated. The host ends the mode by sending a zero-length block. Mstor closes the file, returns an {ACK} character and is available for new commands. The host should implement a timeout to handle incomplete or no-response blocks keeping in mind that if Mstor is power-cycled while in this mode, it returns to operation in Command Mode and will respond to a block request with bad-command {NAK} errors.
&R	**Read File Stream Mode… uses stream protocol.** **Typical command: &Rpath/filename.ext{CR} .** Mstor responds with {ACK} or {NAK}E-xx Immediately following the {ACK} is the total decimal number of bytes to be returned – the filesize -- enclosed in <> brackets. Mstor then waits for group-request characters and delivers one *groupsize* of characters for each request. The host can abort the

	operation by sending NAK as a group request. There is no error checking of the data stream. Once in Read mode, no other files can be opened and no other commands can be initiated. After all data is sent Mstor closes the file and returns an {ACK} . If the host aborts, Mstor returns {NAK}E-xx to indicate that the transfer was incomplete.
&r	**Read File Block Mode… uses block protocol.** **Typical command: &rpath/filename.ext{CR}** . Mstor responds with {ACK} or {NAK}E-xx **.** On {ACK} Mstor then waits for an 'N' character to return the next block. Mstor returns blocks containing *blocksize* total characters except for the last block of file data which may be shorter. Block formatting is: **< C data-chars KK >** ..see above The <> characters are block framing characters. For a retried block the < > characters are replaced with [ ]. C is the binary count of data bytes in the block. The two KK check characters are a high-byte-first character pair representing a checksum – a continuous sum of all characters except the check characters, or a 16-bit CRC. When Mstor reaches the end-of-file and has sent all file data, it responds with a zero-length block. The host returns an {ACK} and Mstor reverts to Command Mode to accept new commands. (As with other blocks, the host can request a retry of the zero-length block if it fails to validate). The host can abort at any time by returning NAK as a request character causing Mstor to return {NAK}E-xx and exit the mode.
&S	**Seek File Stream Mode… uses stream protocol** **Typical command: &Soffset>path/filename.ext{CR} .** This mode is identical to the Read File Stream Mode above except that instead of starting from the beginning of the file, reading starts at the byte offset specified as a decimal number of 1 to 10 digits. Mstor responds with ACK<bytes-to-transfer> or {NAK}E-xx if the file is not found or the offset is larger than the file (<bytes-to-transfer> is the difference between the offset provided and the size of the file so the host should expect this total number of bytes). There is no error checking. Mstor returns one *groupsize* sequence of file characters in response to each group-request character. Once in Seek mode, no other files can be opened and no other commands can be initiated. When Mstor reaches end-of-file it responds with {ACK}, closes the file and returns to Command Mode. During receipt the host can abort by issuing an 'X' as the group-request character. Mstor returns {NAK}E-xx and exits the mode.
&s	**Seek File Block Mode… uses block protocol.** **Typical command: &soffset>path/filename.ext{CR}** . This mode is identical to the Read File Block Mode above except that reading begins at the specified byte offset which is a 1 to 10 digit decimal number. Mstor responds with {ACK} or {NAK}E-xx . Mstor then waits for an 'N' character to return the next block. Refer to the &r command above for further details.

&A	**Append File Stream Mode… uses stream protocol.** **Typical command: &Apath/filename.ext{CR}** . Identical to the &W command above except that the file is NOT overwritten, it is appended. If the file does not exist it is created.
&a	**Append File Block Mode… uses block protocol.** **Typical command: &apath/filename.ext{CR}** . Identical to the &w command above except that an existing fille will be appended rather than being overwritten.
&L	**List Files (Stream Mode) .. uses stream protocol.** **Typical command: &Lpath/mask{CR}** . Mstor returns {ACK} if the directory is found and mask is valid, or {NAK}E-xx . The file list is streamed in groups of characters as controlled by *groupsize* as the host repeatedly requests groups. The host can abort the streaming file list by sending NAK as the request character. Any other character returns the next group of file list characters. File 'items' are returned as described above and when the last character of a group is {ACK} the data has ended. See above for the format of the file-items returned. Wildcards '*' and '?' can be included in the mask. If no path and no mask is provided a default mask of *.* is assumed. If a path is included in the command, a mask MUST also be included.
&D	**List Directories (Stream Mode) .. uses stream protocol.** **Typical command: &Dpath/mask{CR} .** Mstor returns an {ACK} for a valid path or {NAK}E-xx . It does NOT descend down into subsequent subdirs to list any deeper subdirectories, so does not create a descending tree. It lists only the subdirs immediately under the specified directory/path. The list is streamed in groups of characters as controlled by *groupsize* as the host repeatedly requests groups. The host can abort the streaming file list by sending NAK as the request character. Any other character returns the next group of characters. Directory 'items' are returned as described above and when the last character of a group is {ACK} the data has ended. See above for the format of the directory items returned. Wildcards '*' and '?' can be included in the mask. If no path and no mask is provided a default mask of *.* is assumed. If a path is included in the command, a mask MUST also be included.
&E	**Set/Get END_OF_SEQUENCE string… uses no protocol.** **Typical command: &E{CR} or &EABC?l0-%ne\9d+@{CR} .** Mstor returns {ACK} or {NAK}e-xx and with no parameter included in the command, returns the current 16-byte EOS. With a 16-byte parameter it sets Mstor's EOS bytes and saves it in non-volatile memory. The host should retain and use this value to terminate its write-streams. When in the streaming Write/Append modes, Mstor continually monitors the incoming data stream for this sequence of bytes. When detected (and followed by no additional bytes for 250ms), Mstor recognizes that the host is done streaming, so terminates the mode. The Mstor default 16-character EOS array: **eos_array[16] = { 0x41, 0x02, 0xFE, 0x33, 0x17, 0x99, 0x80, 0x12, 0x53, 0xCD, 0xA7, 0x09, 0xB6, 0xE3, 0x72, 0x68}**

	Streaming Write/Append are the only modes which use the EOS array. Note that this array is NOT terminated with a null, but is referred to as a string also.
&g	**Get/Set Group Size. .. uses no protocol** **Typical command: &g32{CR}** . With no parameter Mstor returns the current *groupsize* as a 2/ 3 digit decimal number. With a 2/3 digit decimal number parameter (16 to 250) it sets the *groupsize*. *Groupsize* is used in streaming transfers and defines the number of bytes returned in response to each single-character request from the receiver. This allows allows communications with limited-size buffers, and provides the receiver time to process that data, before requesting another group. Mstor issues {ACK} on success or {NAK}E-xx where xx is a two-decimal-digit error code. (Host must have a serial receive buffer of at least 16 characters).
&N	**Rename a File. .. uses no protocol.** **Typical command: &Npath/filename.ext> newname.ext{CR} .** Mstor returns {ACK} on success or {NAK} E-xx . Note that the existing file may be preceded with a directory path string, but newname must not. The newly named file will remain where the original was. Filenames must confirm to 8.3 format.
&F	**Delete a File… uses no protocol** **Typical command &Fpath/filename.ext{CR} .** Mstor returns {ACK} on success or {NAK}E-xx The file can be above or below the current directory in the directory tree and its location is specified with the optional path string.
&B	**Set/Get the BlockSize for Block Transfers… uses no protocol** **Typical command &B35{CR} .** Mstor returns {ACK} on success or {NAK}E-xx where xx is a two-decimal-digit error code.. With no parameter it returns the current *blocksize*, and with a 2/3 decimal digit parameter sets to 16 – 250 and saves to non-volatile memory. The smallest blocks incur significant framing overhead during block-mode transfers, but can be useful in systems with limited resources.
&b	**Set/Get Block Timeout… uses no protocol** **Typical command: &b10{CR}** Mstor returns {ACK} on success or {NAK}E-xx where xx is a two-decimal-digit error code. With no parameters it returns the current block-timeout. With a parameter of 1-254 it sets the block-timeout in 100ms increments to 0.1 sec to 25.4 sec. A setting of 255 defines an indefinite block timeout. During block mode writes, if an Mstor-requested block is not fully received within this time, Mstor requests a retry of the block. 3 consecutive block timeouts ends the mode and returns an error. The host should implement a similar timeout and block-retry mechanism as outline in the sample code provided for block mode reads.
&o	**Set/Get Stream Timeout… uses no protocol** **Typical command: &o20{CR}** With no parameters it returns the current stream-timeout. With a parameter of 1-254 it sets the stream-timeout to 0.1sec to 25.4 seconds.

	A value of 255 sets an indefinite stream timeout defeating the feature. During stream mode writes, if an Mstor-requested group is not received within this time, Mstor terminates the mode and returns {NAK}E-xx . During stream mode reads, if Mstor does not receive a group-request for this period of time it ends the mode and returns {NAK}E-xx.
&C	**Change Directory .. uses no protocol. – move up or down in the directory tree. Typical command: &Cmydir/nextdir/** . This will move Mstor down into mydir, then into nextdir below *the current directory*. **&C/mydir/nextdir** modes two levels below the root directory because the leading / references the root. Mstor allows the use of either foreslashes, '/' or backslashes '\' in path strings. Mstor responds with {ACK} if the command was successful, or {NAK} followed by E-xx where xx is a two decimal digit error-code. To move up in the directory structure, use the .. (two dot shortcut) format which moves up one level in the directory tree. Ex: & **C….{CR}** moves two levels up**.** The host must have knowledge of the directory structure in order to successfully move within it.
&M	**Make New Sub Directory… uses no protocol** **Typical command: &M/path/newdir{CR} .** Mstor returns {ACK} on success or {NAK}E-xx on error. . The new directory must not already exist. All directory names must must be in 8.3 format. Path is optional and can be above the current directory level, or below it in a different branch such as &M…./brnch2/newdir.abc
&K	**Delete a Sub Directory… uses no protocol** **Typical command: K[MODE]path/dirname** Mstor returns {ACK} on success or {NAK}E-xx . Note that this command requires a mode of operation as follows: [MODE] is one of two characters specifying : **P** (which defines Protected Mode) The directory will ONLY be deleted if it is empty. **U** (which defines Unprotected Mode). The directory and all contained files and subdirectories will be unconditionally deleted. All directories must be in 8.3 format. Note that the subdirectory can be above or below the current level using the appropriate path. Refer to the &C command for path string info. The root directory cannot be deleted.
&T	**Get/Set Real-Time Clock (12 Hr format)… uses stream protocol** **Typical commands: &T{CR} ..** Mstor returns {ACK} and awaits a group request to stream the response using stream protocol**:** YYYY-MM-DD HH:MM:SSxM{ACK} Year is 4 characters, then Month/Day/Hour/Minute/Second are all two characters … x = A or P representing AM or PM. For each request character received Mstor returns one group-size of characters. When the last byte of a group is {ACK} the output is complete. If the host delays longer than Stream-Timeout in its requests, Mstor terminates the command.

	&TYYYY-MM-DD HH:MM:SSxM{CR} . Mstor sets the clock to this time. Return is {ACK} or {NAK}E-xx . Mstor has a large receive buffer and can accept the entire command so no protocol is used. The returned time can be invalid (all zero's) if the clock has not been set or the battery has discharged.
&V	**Get Mstor Code Version… uses no protocol** **Typical command: &V{CR}** Mstor returns {ACK} followed by a string containing the current code version number i.e. 'MSTOR V1.02a'{ACK}. Since this string is always less than 16 characters no protocol is used to return it to the host.
&P	**Get/Set Mstor Power State… uses no protocol Typical Commands:** **1) &P{CR} --** returns {ACK} followed by N = USB power-ON, or F=USB power OFF **2) &PF{CR} –** Power-Down the USB drive. **3) &PN{CR} –** Power-UP the USB drive. **4) &PS{CR} -** Go to low-power Sleep state

To minimize the code required of a small microcontroller, setting the Real-Time clock, block size, block-timeout, group size, stream-timeout and END_OF_SEQUENCE array can all be performed using the Mstor Setup application on a PC since all these values are saved in Mstor's non volatile memory. And if you pre-create the USB disk's directory structure on a PC, then the directory structure creation can be 'outsourced' as well. This minimizes the host computer source code needed to configure and operate the USB disk.

Mstor Hardware Resets:

As an asynchronous device like Mstor sequences through various operations it can get out of sync with its host. Even with Mstor's fail-safe stream and block timers a reset may still be needed to force a return to a known state. Mstor also requires a hard reset to wake up from its low power SLEEP state. A 'hard-reset' can be applied using several methods. Mstor restarts operation in Command Mode with all files closed, and all values set to their most recent user settings – it's power-up state.

1) Issuing an RS232 'break state' for at least 100ms. This is not a single RS232 character but rather an RS232 state that holds the host's TX signal active continually. Many host serial ports provide this capability. Mstor hardware detects this and generates a hardware reset.
2) Setting the RS232 RTS (request to send) signal active for a minimum of 5ms. (Mstor does not use RTS as a serial handshake signal). The RTS signal is internally 'lightly' held at an inactive state so if the host does not drive this signal, it should not be wired in the DB9 cable to prevent noise pickup.
3) Remove and reapply power.

Mstor Power States:

Mstor has 3 power-states in which it can operate:

1) **Fully Powered.** This is the state that Mstor enters when power is first applied, or after any of the above resets are applied. It is fully active, the USB port and any connected drive is fully powered and ready to execute disk access or system configuration commands. Mstor's green LED slows its flash rate once the USB drive is initialized and ready. (Mstor requires approximately 35mA, and a USB drive typically requires an additional 30-80mA).

2) **USB Powered-Down.** Mstor is operational for non-disk access commands (such as configuration commands), but the USB drive power is turned off reducing overall power consumption to about 50% of full power. Mstor's green power LED flashes very quickly indicating that no USB drive is detected. The &PN command (see above) is available to re-power the drive and return to the fully powered state in the few seconds required to enumerate the drive (re-initialize USB communications). And the &PS command is available to put Mstor into an extremely low-power sleep state.

3) **Sleep State.** Mstor is completely shut down except for the very low power real-time clock that remains running. Mstor will not respond to serial commands, no LEDs light and the USB drive is powered down. While power is applied, the microcmps of power used are supplied from the power supply. If power is removed, the internal battery continues to operate the real-time clock for up to 3 months. When power returns Mstor returns to full powered operation and recharges its internal battery. In this state Mstor can be reset using the RTS signal, or by removing and re-applying power.

Timekeeping and File TimeStamps:

Mstor provides a battery-backed real-time clock for time- stamping files when they are created and updated. During manufacture and testing each Mstor internal clock is set to Pacific Standard Time. It is accurate to a few seconds per month and the internal battery will keep this clock active for 3 months or more in the sleep state. The internal battery recharges when power is applied. If powered down for an extended period and the battery discharges, the clock will need to be reset. (See &T command above) . Mstor time is readable and can act as the real-time clock reference for your system.

Reliability Features:

Mstor implements an internal watchdog timer, and a power-level monitor. If the applied power drops to 4.5V Mstor aborts any disk operations in progress and enters sleep mode – everything shuts down except the real-time clock which is powered by the battery. When valid power returns (and transitions to an operational level in 100ms or less) Mstor automatically resets and resumes operation.

RS232 Baud Rate Control:

The RS232 baud rate is set using the 3 internal shorting jumpers as shown below in the Table. To access these jumpers the case must be opened. Very slow baud rates are included so Mstor can be installed at a distant location. Parity is not supported since the block-transfer modes provide much better error checking. The table shows shaded blocks where shunts are to be installed.

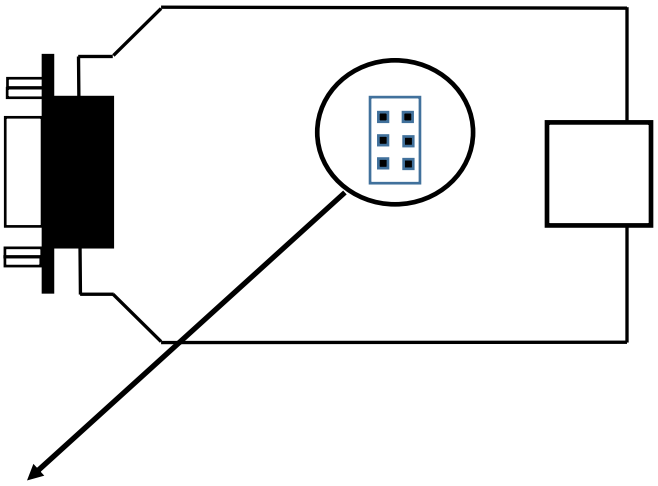


DIAGRAM	RS232 BAUD RATE
	1200 bps
	4800bps

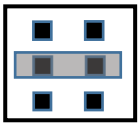
	9600 bps
	19.2k bps
	38.4k bps
	57.6kbps
	115k bps
	250k bps

TABLE 1

.

Mstor devices are configured for 9600 baud when shipped.

RS-232 Signal Compatibility:

Mstor is compatible with standard RS-232 signals levels which go both positive and negative (above and below 0 volts) and is also compatible with the 0-5 volt signal levels used in some systems. The ST202 RS-232 driver has a receive threshold at +1.2 volts above ground so even signals that do not go below ground operate reliably.

Power Application:

Mstor requires +5VDC internally to operate and it provides this power to the USB port for the USB drive. There are two ways to apply power Mstor:

1) Apply 5VDC to a pin 4 or 6 of the DB9 connector depending on model.
2) Apply 6-9VDC power to the power jack from a wall-adapter.

Mstor Error Codes:

NO_ERROR	0
BAD_COMMAND	1
FILE_NOT_FOUND	2
DIRECTORY_NOT_FOUND	3
FAILED_TO_OPEN_FILE	4
FAILED_TO_CLOSE_FILE	5
INVALID_PARAMETER	6
INVALID_COMMAND_LENGTH	7
FILE_WRITE_ERROR	8
WRITE_TIMEOUT_OCCURRED	9
DIR_PATH_ERROR	10
UNKNOWN_COMMAND_STATE	11
FAILED_TO_CREATE_DIRECTORY	12
BAD_MASK	13
FAILED_TO_DELETE_FILE	14
FAILED_TO_DELETE	15
INVALID_COMMAND_MODE	16
SEEK_ERROR	17
TOO_MANY_CONSECUTIVE_RETRIES	18
INVALID_8P3NAME	19
WILDCARDS_NOT_ALLOWED	20
SEEK_POSITION_ERROR	21
GENERAL_SEEK_ERROR	22
TIMEOUT_OR_USER_ABORT	24
FAILED_TO_RENAME_FILE	25
FILE_WRITE_FAILURE	26
STREAM_TIMEOUT	27
COMMAND_TOO_LONG	28
INVALID_DECIMAL_POSITION	29
INVALID_8P3_CHARACTER	30
FNAME_TOO_LONG	31
DISK_NOT_READY	32
INVALID_BLOCK	33
GENERAL_ERROR	34
SOURCE_NOT_FOUND	35
USER_ABORT	36
BLOCK_COMMAND_TIMEOUT	37
INVALID_RETRY_REQUEST	38

Mstor MODELS AVAILABLE

Model#	Features
Mstor-014	(DCE) 5VDC power on pin 4 of DB9
Mstor-016	(DCE) 5VDC power on pin 6 of DB9
Mstor-034	(DTE) 5VDC power on pin 4 of DB9
Mstor-036	(DTE) 5VDC power on pin 6 of DB9
Mstor-024	(DCE) 6-9 VDC power from wall-brick
Mstor-044	(DTE) 6-9 VDC power from wall-brick

TABLE 2

Power Considerations:

USB ports are sometimes used for battery chargers, lighting power sources or powering external rotating-disk USB drives. The Mstor USB port should ***not be used for any of these devices***. Typical solid-state USB 'thumb' drives require 40-80ma, and while standard USB 2.0 ports can provide up to 500ma for device operation or charging, Mstor is limited to USB drives that require no more than 100ma.

Mstor Physical, Electrical, Environmental Specifications

Physical:

Size:	(2.9" x 1.7" x 0.9" not including USB drive
Weight:	3oz
USB Connector:	Standard Type-A
Host Connector:	DB9 female
Case Color:	Black
Power Jack Connector:	5mm X 2.1 mm (Ctr Positive)

Electrical:

Absolute Maximum Input Voltage	+15VDC (Internal Regulator Models only)*
Maximum USB Drive Current	100mA
Nominal Voltage Input:	+5VDC +/- 5% or 6-9VDC (per model number)
Current Input:	Typical 35mA plus USB drive current
Baud Rate (RS232 port):	1200 to 115k selectable
RS232 Signal Inputs	+/- 25V absolute maximum HIGH threshold 2.4V minimum LOW threshold 0.8V maximum

Environmental:

Max Operating Temperature:	+70ºC
Min Operating Temperature:	10ºC
Max Storage Temperature:	+70ºC
Min Storage Temperature:	-10ºC
Humidity:	Non-condensing at all temperatures

Document Revision Record:

Revision	Revision Date	Description
V1.0x	Apr 17, 2022	Preliminary
V1.0x	June 21, 2022	Preliminary. Previous version streaming modes returned characters in a continuous stream using Now stream-mode uses *groupsiz*e groups of characters in request/response format.