Versalent

www.versalent.biz

RS232 INTERFACE TO MASS STORAGE

Mstor Manual

Preliminary -- Version 1.0x
Revised Apr 17, 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. Data transfer speeds are slow compared to SCSI or SATA connections, but systems which have no disk interface now have a means of accessing large storage -- it is intended for use in any system where fast transfer speed is not a requirement.

Mstor communicates through an RS232 interface using simple one-character commands. Data can be transferred as a ***stream of bytes*** for the simplest minimal code-space implementation, or using a ***block-framed format*** with check characters for improved data integrity. Block sizes can be set to (default) 16 bytes, or 32/ 64/128 bytes for more efficient transfers Check characters are by default a simple 16-bit checksum (requiring minimal host processing power) or an optional 16-bit CRC. Block transfers require the receiver to request each block so data is inherently 'throttled' by those requests. Streaming transfers use XON/{XOFF}serial protocol to prevent receiver buffers from being overrun.

The FAT formatted USB drive is compatible with PC drives so you can access/edit/transfer files 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.***

Block transfers add 5 framing bytes to the 16/32/64/128 data bytes-per-frame resulting in frames of 21/37/69/133 total bytes. The smaller sizes are less efficient due to the framing bytes but are provided for small systems with limited resources. By default, blocks use a (2 byte) 16-bit Checksum for block validation – or a 16-bit CRC can be enabled.

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 always shown 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, or 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.

File System Limitations:

Mstor allows wildcard masks only with the List Files (&L), and List Directories (&D) commands. This means that Mstor cannot operate on batches of files .. 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.

Detailed Operation:

For disk read /write/append operations, the host uses single-character streaming mode commands (&R, &W, &A) or block mode (&r, &w, &a) commands. Mstor first responds with {ACK}* or {NAK}* indicating command acceptance. After an {ACK} the transfer begins, or after a {NAK}, an error indicator E-xx is returned (xx is a two-decimal-digit error code).

*{ACK}=0x06 , {NAK} =0x15 are ASCII characters noting command acceptance/rejection.

Streaming Mode Transfers:
In streaming modes there are no handshake exchanges between sender and receiver … and no error checking .. the sender delivers a continuous stream of characters which can be paused by the receiver issuing {XOFF}* then {XON}* to 'throttle' the transfer if necessary. Both read and write operations have built in lockup-prevention as described below.

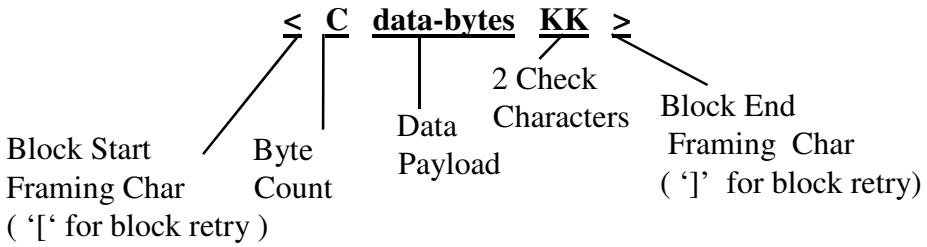
*{XOFF} is the ASCII character = 0x13, {XON} is the ASCII character = 0x11

Block Mode Transfers:

In block modes, the receiver requests each block with a single character -- 'N' for next block, X' to abort the command, or *any other character* to retry the last block. {XOFF/XON} is not used since data-blocking inherently throttles the data. (The receiver checks each returned block – if framed with < > characters, it is the response to a 'N'ext block. If framed with [ ] characters it is the response to a retried block. This scheme allows both sender and receiver to manage blocks even if the request characters get corrupted.

More details on these transfer modes follow.

Block Format:



If the block is a 'retry' block 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 of 16/32/64/128. 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).. The high byte is first.

< > or [ ] are the block framing characters. Most blocks are framed with **< >** however if the receiver fails to validate a block's check characters, its next request should NOT be 'N' to request the next block. Any character except 'N' or 'X' will request a re-transmission of the last block. This system allows the receiver to handle 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.

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 ends the process by sending the predefined END_OF_SEQUENCE string which is not stored to disk. If stream-timeout is not set to 'indefinite period', and this time passes with no characters received, Mstor automatically ends the mode, closes the file, returns a {NAK} and re-enters Command Mode. The stream-timeout can be user set from 1 to 60 seconds (default = 5), or to 255 which sets the timeout period to 'indefinite' (Refer to the 'O' command)..

   The host should monitor for any returned messages during this mode – it should receive none. However if Mstor undergoes a power-reset for instance, it will power-up in Command Mode .. and will return {NAK}'s in response to incoming data characters – the host should take appropriate action.

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 it sending 'N' for next-block, or any other character for a block retry, or an 'X' to abort the mode. After issuing the block, request Mstor waits for up to the block-timeout period for a block to start. The 2nd character received is the block length, so when this number of characters has been received Mstor processes the block. If validated, the next block is requested.
   If a block is not validated, Mstor flushes any further characters received and requests a retry of the last block. If too few characters are received within the block-timeout period after an Mstor request, anything received is flushed and a retry is requested. ***(The host must NOT send any un-solicited blocks/characters during this process. Mstor is in control and the host must follow its instructions.)*** When the host is done and has no more data to send, or it wants to abort for any other reason, it sends an empty block containing zero data bytes . Mstor responds with an {ACK} (or a retry if this block arrives corrupted). The host must return another empty block … until it receives an {ACK} indicating that the command is complete . Mstor is then in 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> then begins streaming the data. *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. It will accept {XOFF/XON} to pause/resume the continuous stream or 'X' to abort the process. Anything else will be ignored. It remains in this mode until it streams the *filesize* number of bytes reported or the host aborts the process. It then issues an {ACK} and returns to Command Mode. Note if the host leaves Mstor in {XOFF} state for more than the stream-timeout period, Mstor automatically resumes transmission to complete the process – preventing permanent lockup.

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 requests for each block. 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 cannot be found, Mstor returns {NAK} and an error code. If a received block fails to validate, the host must request a retry (any character other than an 'N' or 'X'). The host must implement its own timeouts as necessary to request a retry if an incomplete block is received after a period of time. (2nd byte of each block specifies the number of data bytes contained). Mstor signals the end of data by sending a zero-byte block. The host can abort the process by returning an 'X' instead of a block request. Sample code available at www.versalent.biz/downloads/Mstorcode.c

6) **List Stream Mode** -- Mstor has received a stream List Files command (&L) and is in the process of streaming a list of file items to the host. In this mode Mstor responds to {XON/XOFF} to pause transmission, and 'X' to abort the process. A path can be included such as &L/mydir/nextdir .The list contains all the files 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 M/D/YYYY format
Create-time is H:MM:AM format
Size is number-bytes, or if > 999, number kbytes i.e. 23k
{CR}{LF} newline separator to support display on a screen

7) **Directory Stream Mode** – Mstor has received a stream Directory List command (&D) and is in the process of streaming a list of sub-directories items at this level in

the tree. In this mode Mstor responds to {XON/XOFF} to pause transmission, and 'X' to abort the process. The list contains the same item format as above for file item format.

Mstor Transfer Timeouts:

Transfer timeouts provide for recovery from unexpected states. If Mstor is waiting for streaming or framed data that does not arrive after the predefined timeout period, Mstor recovers. There are two timeouts – one for streaming transfers, and one for block transfers. Each can be set from 1 second to 60 seconds, or indefinite. The default transfer timeouts are 5 seconds.

Stream Timeout – This timeout is used in streaming modes only. 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} followed by E-xx (xx = two decimal-digit error code), and returns to Command Mode. While Mstor is reading a file and sending a stream, if it receives an {XOFF} but does not receive an {XON} for the stream-timeout period, it generates its own {XON} and resumes sending. These mechanisms prevent indefinite lock-up of a streaming transfer.

Block Timeout -- This timeout is used in block modes only. 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 and resets the block timer. It will re-issue this request a maximum of 5 consecutive times before it aborts the mode, closes the file, responds with an error and returns to Command Mode.
When Mstor is reading the disk and responding to block requests from the host, the block timeout is not used. Mstor will wait indefinitely for the host to request a block, or terminate the mode.

Mstor End_Of_Sequence string:

When **writing/appending data in streaming mode**,**only** .. the data stream cannot inform Mstor when end-of-data occurs using any single character since all are valid file characters. If the host simply stops sending Mstor cannot determine if this is temporary and data will resume or if this is the end of data.
To inform Mstor when it is done, the host sends the additional predefined **End-Of-Sequence** string – a sequence of 16 characters that Mstor recognizes as the end of data. This **End_Of_Sequence (EOS)** string should be a sequence that is unlikely to 'accidently' appear in the data stream. It should not contain sequential characters like ABC…or 567…. A more random mix of 8-bit values is less likely to exist in file data. When Mstor recognizes this string it also verifies that nothing follows for a period of 250ms. See the &E command below to use the default Mstor sequence, or create your own – this string is stored in non-volatile memory.

Streaming read commands do NOT use End-Of-Sequence because Mstor responses start with the number of bytes to be returned (the file size). The response starts with an {ACK} < byte-count > … and ends when Mstor has sent that number of bytes. Block transfers also do not use EOS since a zero-length block is sufficient 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 without catastrophic effect. For data that must be accurate, the block modes are provided. Data is transferred in framed data blocks that contain check characters as checksums or CRC's and allow the receiver to request retransmission of any received block whose computed check characters do not match those received in the block.

Mstor Data Transfer Speed:

Transfer speeds are limited by the serial port speed. At 9600 baud Mstor can stream 1k bytes per second. At 115k baud it can stream 11k bytes per second. (Streaming commands are &R, &W, &A). Blocked (checksum/CRC verified &r, &w, &a modes) transfer rates are slower since it is a hand-shake process of request/transfer/repeat …. In addition, the receiver requires a slight delay to validate the check characters before deciding whether to request the 'N'ext block, or to ask for a retry of the last block, and there is some overhead incurred with the framing bytes. Block transfers are typically 40-60% slower than streaming transfers however when data integrity is more important than speed, block transfers are preferred. The low baud rate of 1200 is provided for the longest distance transmission (120 bytes/sec at over 200 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 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 the host (or a PC) builds a complex, multi-branch directory structure on the drive, the host must retain knowledge of that structure in order to navigate the tree. Without the visual tree structure of a PC it is suggested that the tree structure be kept as simple as possible.

Mstor Commands:

Commands consist of the ampersand '&' character followed by a command byte, followed by a command parameter(s), and end with a 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. Every command results in a response even if it is just an acknowledge.
In the table of commands below, commands which set Mstor in an altered mode are tinted. Commands are all ASCII characters. Mstor generally responds with an ASCII {ACK} (0x06) or {NAK}(0x15) sometimes followed by additional information. Path in disk-access commands is optional. {NAK}E-xx is a {NAK} followed by E-xx with xx being a two-decimal-digit error code.

Mstor Command List

Command	Command Description
&W	**Write File Stream Mode.** **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. After that all serial characters sent to Mstor get written to this file until the END_OF_SEQUENCE (EOS) string is received. There is no error checking. The file remains open for data pauses of up to the stream-timeout period (which can be indefinite). This mode ends when Mstor receives EOS followed by 250ms of 'no-data', or there is a data pause for more than the stream-timeout period. Mstor responds with {ACK} for an EOS ending, or {NAK}E-xx for a stream-timeout ending. Mstor issues the response, closes the file and returns to Command Mode. During write streaming, if Mstor returns {XOFF} to the host it must pause its transmission until Mstor issues an {XON} to resume (or the host times-out the {XOFF} state). The stream timeout is extended by an {XOFF} period.
&w	**Write File Block Mode.** **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 of the current BLOCKSIZE …16/32/64/128 framed data bytes. After each validated block Mstor sends 'N' for 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 >** The < > characters are block framing characters. For a retry block the < > frame characters are replaced with [ ]. C is the binary byte count of the data bytes in frame. 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 BLOCKSIZE + 5

	characters in each framed block so 16/32/64/128 byte blocks contain 21/37/69/133 bytes per frame. 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. {XOFF} is not used in this mode since Mstor will simply delay its request for the next block if it needed. The host should implement timeouts to handle incomplete or no-response blocks keeping in mind that if Mstor is power-cycled during this process, it returns to operation in Command Mode and will respond to block requests with bad-command {NAK}s.
&R	**Read File Stream Mode.** **Typical command: &Rpath/filename.ext{CR} .** Mstor responds with {ACK} or {NAK}E-xx Immediately following the {ACK} is decimal number of bytes to be returned enclosed in <> brackets followed by a stream of data bytes. The response looks like ACK<81325>data bytes….{ACK} . ({ACK} = 0x06, number data bytes = 81325) Mstor streams the file contents pausing if the host sends an XOFF, then resuming on XON. Mstor returns the entire file only pausing if {XOFF}'d by the host. The host can also abort the operation by sending a single 'X' character at any time. There is no error checking. Once in Read mode, no other files can be opened and no other commands can be initiated. After all data is sent Mstor stops sending, closes the file and returns an {ACK} . If the host aborts Mstor returns {NAK}E-xx to indicate that the command was incomplete. Note that if Mstor had been {XOFF}'d it expects an {XON} to resume communications. If it does not then receive an {XON} within the stream-timeout period it resumes communications. This provides automatic recovery from sequencing errors.
&r	**Read File Block Mode.** **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. The default blocksize is 16 data characters, but can be set to 32/64/128 bytes see &B command. Block formatting is: **< C data-chars KK >** The <> characters are block framing characters. For a retried block the < > framing characters are replaced with [ ]. C is the binary count of data bytes in the frame. 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. So there are BLOCKSIZE + 5 characters in each framed block so 16/32/64/128 data-byte blocks contain 21/37/69/133 total bytes per frame. In block modes, {XOFF} is not active since the host will simply delay its request for the next block if needed. When Mstor reaches the end-of-file it returns a zero-length block, sends an {ACK}, and reverts to command

	mode to accept new commands. If a host abort ('X') causes the termination, Mstor returns {NAK}E-xx and ends the mode.
&S	**Seek File Stream Mode. . 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. If {ACK}, the file will be returned as a continuous stream of serial characters. <bytes-to-transfer> is the difference between the offset provided and the size of the file so the host should expect this number of bytes. The host can use {XOFF} to pause transmission to prevent host overrun. There is no error checking. Once in Read mode, no other files can be opened and no other commands can be initiated. During receipt the host can issue a single 'X' to abort the command. If aborted, Mstor returns {NAK}E-xx . If Mstops reaches end-of-file it responds with {ACK}, closes the file and returns to Command Mode.
&s	**Seek File Block Mode.** **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 waits for an 'N' character to return the next block. The default blocksize is 16 characters, but can be set to 32/64/128 bytes see B command. Block formatting is: **< C data-chars KK >** The <> characters are block framing characters. Retried blocks replace the < > framing characters with [ ]. C is the 1 binary byte count of data bytes in the frame. The two KK check characters are a high-byte-first pair representing a continuous sum (checksum) of all frame characters except the check characters or a 16-bit CRC. There are BLOCKSIZE + 5 characters in each framed block so 16/32/64/128 byte blocks contain 21/37/69/133 total bytes per frame. {XOFF}is not used since the host can pause transmission by delaying its block request.. The host can abort the command by sending 'X' instead of 'N' which ends the mode, closes the file, and returns an {NAK}E-xx. If Mstor seeks to the end of file it returns a zero-length block to inform the host of end-of-data, returns an {ACK} and enters Command Mode.
&A	**Append File Stream Mode.** **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.** **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	**Set List File (Stream) Mode.**

	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 so the host must use {XOFF}to pause the stream as needed. The host can abort the streaming by sending an 'X' character. Mstor returns an {ACK} at the end of the stream, or {NAK}E-xx if the host aborts the transmission. See above for the format of the file-items returned. Wildcards '*' and '?' can be included in the mask. If no mask is provided it is assumed to be *.* so all files are returned.
&D	**List Directories (Stream) Mode.** **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 so the host must use {XOFF} to pause the stream as needed. The host can abort the streaming by sending an 'X' character. Mstor returns an {ACK} if it completes the list, or {NAK}E-xx if the command is aborted. See above for the format of the directory-items returned.
&E	**Set/Get END_OF_SEQUENCE string.** **Typical command: &E{CR} or &EABC?l0-%ne\9d+@{CR} .** With no parameter it returns the current 16-byte EOS string. With a 16-byte parameter it sets Mstor's EOS string and saves it in non-volatile memory. The host must save this value. 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 100ms), Mstor recognizes that the host is done streaming. It closes the file being written. And returns ACK. The Mstor default 16-character EOS string: **eos_string[17] = { 0x41, 0x02, 0xFE, 0x33, 0x17, 0x99, 0x80, 0x12, 0x53, 0xCD, 0xA7, 0x09, 0xB6, 0xE3, 0x72, 0x68, 0x0}** These are the only modes which use the EOS string.
&G	**Get File Size. Typical command: &Gpath/filename.ext{CR}** . Mstor returns the an ACK<decimal num> on success or {NAK} followed by E-xx where xx is a two-decimal-digit error code. <decimal num> is the size of the file in bytes and can be up to 10 digits. Path can be above or below the current directory – see &C command for path usage.
&N	**Rename a File. 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. 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. Typical command &B64{CR} .** Mstor returns {ACK} on success or {NAK} followed by by E-xx where xx is a two-decimal-digit error code.. Blocksize can be 16/32/64/128 only. The smallest blocks incur significant framing overhead during block-mode transfers, but can be useful in systems with limited resources. Issued with no parameter it returns {ACK} followed by the current blocksize. With a parameter, a new blocksize is set and saved to non-volatile memory. &B{CR} or &B64{CR}
&b	**Set/Get Block Timeout. Typical command: &b10{CR}** With no parameters it returns the current block-timeout. With a parameter of 1-60 it sets the block-timeout to 1 to 60 seconds. A setting of 255 sets an indefinite block timeout. During block mode writes, if an Mstor-requested block is not received within this time, Mstor requests a retry of the block. 5 consecutive block timeouts ends the mode and returns an error.
&C	**Change Directory – mode 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 dots) 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. Typical command: &M/path/newdir{CR} .** Mstor returns {ACK} on success or {NAK} followed by E-xx on error. Xx is a two decimal digit error code. Success or failure. The new directory must not already exist. Newdir 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. Typical command: K[MODE]path/dirname** Mstor returns {ACK} on success or {NAK} followed by E-xx where xx is a two-decimal-digit error code. Note that this command requires a mode of operation as follows: [MODE] is one of two characters specifying : P = Protected Mode. The directory will ONLY be deleted if it is empty. U = Unprotected Mode. The directory and all contained files and subdirectories will be unconditionally deleted. Dirname is a subdirectory name 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	**Set Real-Time Clock. Typical command: &TYYYYMMDDHHMMSSx{CR} .** Mstor returns {ACK} on

	success or {NAK} followed by E-xx where xx is a teo-decimal-digit error code. Month/Day/Hour/Minute/Second are all two character decimal values while year is 4-digit … x = A or P representing AM or PM.
&t	**Get Current Time. Typical command &G{CR} .** Mstor returns {ACK} followed by YYYYMMDDHHMMSSxM{CR} which is the same format as T command above. The returned time can be invalid (all zero's) if he clock has not been set or the battery has discharged.
&V	**Get Mstor Code Version. Typical command: &V{CR}** Mstor returns {ACK} followed by a string containing the current code version number.
&O	**Get/Set Stream Timeout. Typical command: &O7{CR}** Stream timeout can be set from 1 to 60 (seconds) .. or 255 which represents an indefinite time period. During streaming Write/Append transfers the host normally ends the mode by sending the current EOS string which Mstor recognizes and ends the mode. However if EOS never arrives (EOS string is corrupted, or host power-cycles and never ends the stream) .. stream-timeouts provide a fail-safe recovery. If no characters are received for this period of time, Mstor automatically ends the mode, closes the file, and returns a {NAK} followed by E-xx where xx is a two-decimal-digit error code. EOS can be set = 255 defeating this feature, however Mstor could then remain in the Write/Append mode indefinitely. &O{CR} returns {ACK} followed by the current stream timeout, and &O13{CR} returns {ACK} and sets the new timeout to 13 seconds and saves this value to non-volatile memory.
&P	**Get/Set Mstor Power State. 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 time, setting block size, and setting the END_OF_SEQUENCE string commands can all be performed using the Mstor setup application on a PC since these are saved in non volatile storage. And if you create the USB disk's directory structure on a PC, then the directory structure creation can be 'outsourced' as well. This way the amount of small computer source code needed to operate the disk is minimized.

Mstor XON/XOFF and Recovery:

{XON/XOFF} is a useful way to throttle serial communications and prevent buffer overrun and loss of data. {XON/XOFF} is active only during (R command) streaming disk reads, or when the (L)ist file command, or (D)irectory list commands are executed since those can also return a long stream of data. {XOFF} issued by the host can pause the stream and prevent data-overrun in the host's receive buffer.

However using this protocol incurs the possibility of 'lockup' if an {XOFF} has been received, but no {XON} arrives to resume – or the {XON} character gets corrupted. To recover from all possible communications lockup situations, Mstor considers all {XOFF}'s as temporary. That is, if Mstor does not receive an {XON} for a period after an {XOFF}, it clears the {XOFF} condition and resumes communications. This period is 1 second less than the user-setable stream-timeout. This prevents an inadvertent lockup from being permanent. Mstor will resume operation and communication from wherever it was after {XOFF} is canceled.

If the host looses its mode-synchronization and does not get responses it expects, or does not know what mode Mstor is in, or what file is open etc .. it can hard-reset Mstor by issuing an RS232 'break' for 50ms. This is not a single RS232 character. The RS232 break state holds the host's TX signal active continually. Mstor hardware detects this and generates a hardware reset. Any previously opened files are closed and Mstor begins operation in Command mode. Reset is complete when the disk status command (?) return a valid 'disk connected' state.

Mstor Hardware Reset:

To wake Mstor up from its low-power sleep-state, or for a host to recover Mstor to a known state from any command or {XOFF} state or other unknown condition there are several ways to apply a hard reset.

1. Cycling Mstor's power. A reset is applied as Mstor power rises which causes it to re-initialize itself to a fully operational state. If a USB drive is connected, that drive will be detected and Mstor will enter Command Mode – ready to accept user configuration commands and/or disk access commands.

2. RS232-break-character reset. Besides transmitting serial characters, RS232 hosts typically have the ability to place their transmit signal into a 'break' state. This forces the normal TX signal active for an extended period. If Mstor detects the host sending 'break' for more than 20ms, on-board circuits apply a reset to Mstor and remove that reset when the 'break' condition is removed.

3. Mstor does not use the RTS/CTS (request-to-send/clear-to-send hardware handshaking) for transmission 'throttling', so the RTS signal is available as an additional reset input. If RTS is activated for more than 1ms, on-board circuits apply a reset to Mstor and remove that reset when RTS returns to inactive. An unused RTS/reset input can be left unconnected.

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 40-60mA as well).

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 (powered) 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 the USB communications). And the &PS command is available to put Mstor to sleep.

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 remains 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 the battery recharges. Mstor returns to full powered operation when any reset is applied.

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 its internal battery will keep this clock active for 3 months or more in the power-down state. The internal battery recharges when power is applied. If powered down for an extended period the clock may 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 for Mstor installed in distant, perhaps protected locations. Parity is not supported since the block-transfer modes provide better error checking. The table shows shaded blocks where shunts are to be installed.

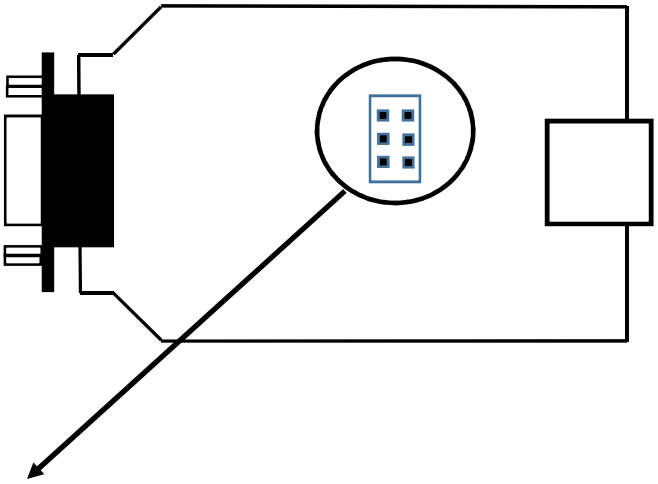
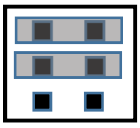


DIAGRAM	RS232 BAUD RATE
	1200 bps
	4800bps

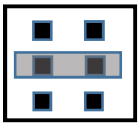
	9600 bps
	19.2k bps
	38.4k bps
	57.6kbps
	115k bps
	250k bps

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 non-standard 0-5 volt signal levels used in some systems. The ST202 RS-232 driver has a receiver 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 MODELS AVAILABLE

Model#	Features
Mstor-5	+5VDC power applied to ribbon connector Pin 3 or Screw Term #2
Mstor-5RD	Same as Mstor-5 with thumb drive recessed into case
Mstor-V	+6-12VDC (variable) power applied to ribbon connector Pin 3 or Screw Term #2, or 6-12VDC center-positive wall supply plugged into Mstor power jack. Internal regulator supplies needed +5VDC.
Mstor-VRD	Same as Mstor-V with thumb drive recessed into case.

TABLE 2

Power Considerations:

USB ports are sometimes used for battery chargers, or lighting power sources or powering external rotating-disk USB drives, however the Mstor USB port should ***not*** be used for any of these types of 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