Versalent

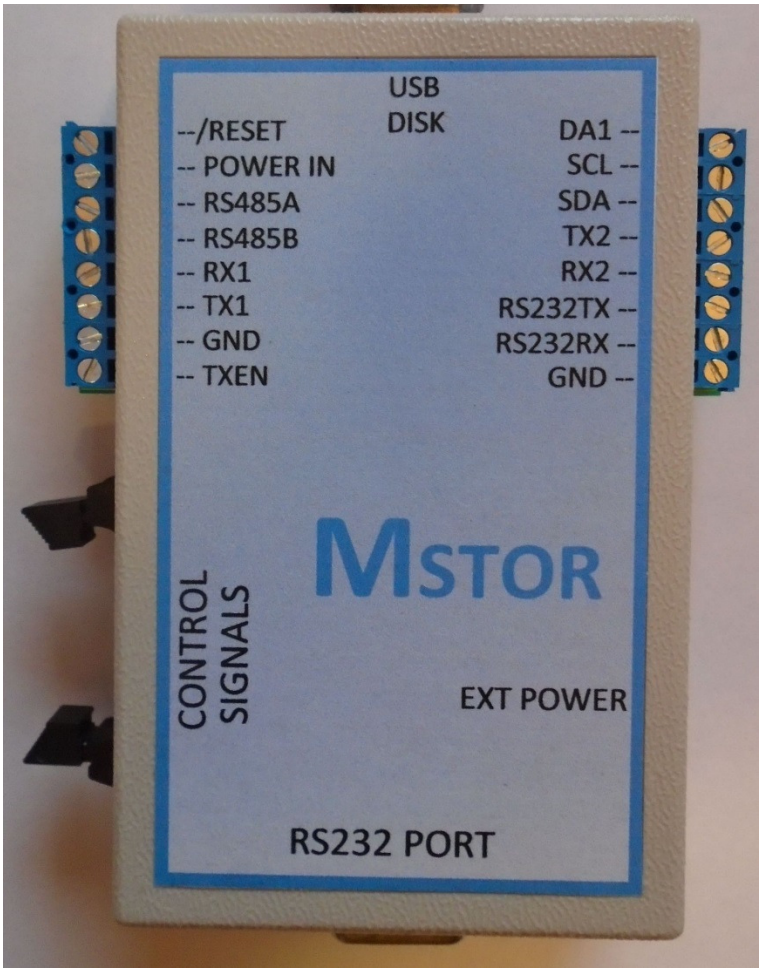
www.versalent.biz

PRELIMINARY **October 2018**

MSTOR USB INTERFACE TO MASS STORAGE FOR MICROCONTROLLERS

MSTOR Manual

Version 1.00
Revised Oct 8, 2018



General Description

MSTOR is a small module (3.8" x 2.4" x 0.9") that allows a standard USB (thumb) drive of up to 32 gigabytes to be used with non-USB microcontroller systems. MSTOR acts as a USB host to automatically enumerate (connect to) a drive, and handle the standard FAT file system low level commands to allow even the smallest of microcontrollers to use mass storage. Data transfer speeds are much lower than systems that offer a more direct CPU-to-drive connection, but systems which would otherwise not be able to access mass storage now have a low cost means of accessing large amounts of data. MSTOR provides an extremely low power, battery backed real-time clock so that all file write operations store and maintain accurate file time stamps.

MSTOR communicates to your system using industry standard interfaces including I2C and RS232/485 . I2C is useful for very short distance communications while RS232/485 can extend the distance of MSTOR from your system to thousands of feet at the expense of data speed. The RS232 port is a configuration port as well as one of 3 available disk-data transfer ports. It communicates to a companion PC configuration application to set the real-time clock, choose the desired disk-data transfer port set RS485 baud rate and/or set the I2C slave address.

The FAT formatted USB drive is compatible with drives used on your PC, so you can use it to transfer files back and forth between a PC and your micro-system. If the host system stores sensitive information that must be secured, you can read/write files using an industry standard AES encryption scheme. By using the RS232 or RS485 interfaces, MSTOR along with its USB drive can be located in a distant, secure location even if the main micro system cannot. The USB drive is external to MSTOR so it is easily accessible allowing for removal and transfer to a PC. Filenames and directories must be in the 8.3 format so to prevent file and directory names from being truncated you should use only 8.3 formatted names.

Disk data is transferred between the host and MSTOR in blocks of 16 to 256 bytes (not including the 5 bytes of framing with CRCs). So the formatted/CRC'd blocks containing 32/64/128/256 data bytes are 37/69/133/261 bytes in total. The small sizes are especially important for small systems that do not have large data buffers although the 5 byte overhead is more significant for smaller blocks. Block structure is *extremely* simple to keep byte overhead low. Disk data transfers are **<[L] …data … [CRC]>** framed and hand-shaked (refer to the Read command below) while other commands that return small amounts of non-disk data use an unframed format. See each command description for details. While disk data blocks from MSTOR are *always* CRC'd , it is up to the host to validate or ignore this CRC. When reading a file, MSTOR waits after retrieving disk data and sending the block to the host. The host must send a signal requesting the next block. When writing a file the host must wait for an MSTOR signal requesting each new block. This allows the block receiver time for each block to be error-checked or processed or written to disk etc. When reading and writing, user programmable block timeouts from 10ms to 30 seconds are also in effect so that MSTOR never waits indefinitely (with a file

open and unavailable to accept a new command). This scheme allows small micros with limited buffers, time to transfer data in/out, and time to compute and validate a CRC if desired, yet provides for recovery from unexpected conditions.

1. **When the Host is Writing Data to MSTOR/Disk :** the user-programmed ***WriteTimeoutValue*** is in effect. When MSTOR has sent an 'N' sub-command back to the host to get the (N)ext block, and the host does not *finish* sending a complete block within this time, MSTOR assumes something went wrong. It aborts the current command, returns an Error code, closes any open file and returns its command processor to an idle state ready for the next command. When the host sets this value, it must allow enough time for the transmission of the N character, then a full 37/69/133/261 byte block plus the maximum host processing time.
2. **When the Host is Reading Data from MSTOR/Disk:** the user-programmed ***ReadTimeoutValue*** is in effect. When MSTOR has queued a block for transmission and is awaiting an 'N' sub command to send the (N)ext block, this user programmable timeout is started. If the host does not respond within this time, MSTOR again times out and takes the same actions as above. When the host sets this value, it must allow enough time for the transmission by MSTOR of a full 37/69/133/261 byte block, plus the host processing time, plus the time for the return of a one-character response (typically 'N').

   Since these timeouts provide fail-safe recovery, the host should not try to implement them precisely. Once calculated it is reasonable to double or triple them to allow for all possible unknown delays which may occasionally occur in reception and processing on either end. (MSTOR uses a 32MHz 16-bit MPU so transmission time dominates its delays).

Also note that COMMANDS MAY NOT BE NESTED. The somewhat limited memory resources of MSTOR means that you may not operate on multiple files simultaneously as with a PC. Once you open any file for Read/Write or Append, you must complete the operation on that file and MSTOR must close it before operating on another file or executing any other MSTOR command. You can simulate simultaneous multiple file operations. For example: open File1 for reading, read/buffer some bytes into the host, abort the read operation, then open File2 for writing or appending, write those bytes and terminate the write operation. Repeat as necessary to transfer data between multiple files. This is simulated multi-file operation because -- ***only one file may be open at a time.***

The available MSTOR commands are:

File And Directory Commands		
Command Name	**Description**	**Command**
List Files/Directories	List files or directories or both w/wo masks	L
Change Directory	Move up or down in the directory structure	D
Make Directory	Create a new Subdirectory in the current dir	M
Delete Directory	Delete a subdir from current dir (specify Protected mode for -- only if empty)	K
Rename File	Rename a file that resides in current directory	N
Delete File	Delete a file that resides in current directory	E
Read File	Read a file starting at beginning	R
Write File	Write a new file, or overwrite an existing one	W
Append to File	Append to an existing file	A
Seek File	Read file from a specified byte-position	F
Get File Size	Get the file size in bytes	S
Encrypt Existing File	Encrypt a file that is in the current directory	Q
Decrypt Existing File	Decrypt a file that is in the current directory	U
MSTOR Configuration Commands		
Interface Select*	Set RS232/485, or I2C as comm method	I
Set Data Block Size*	Set 32/64/128 bytes as max data block size	Z
Set RW Timeouts*	Set READ and WRITE disk timeouts	H
Set RS485 Baud Rate*	RS485 port baud rate	B
Set Encryption ON/OFF*	Set Disk encryption ON/OFF	J
Save Config Values	Save * values above to non-volatile memory	C
Set Time**	Set real-time clock	T
Set Encryption Key***	Set 16 character encrypt key for 128-bit AES	Y
Get time	Get date/time from real-time clock	G
Power Up/Down Drive	Turn drive power ON/OFF, or Put MSTOR to Sleep (<10uA power drain)	P
Get Code Version	Return the version of MSTOR code	V
Abort Command	Flush command buffer	&

TABLE 1 - COMMAND LIST

*Indicates configuration value stored in non-volatile memory so it is retained during power-down and automatically restored at power-up allowing previous operation to resume without re-configuration

** Real Time Clock non-volatility provided by internal battery.

*** Indicates configuration value NOT stored in non-volatile memory to maintain data security

General Command Description/Operation

MSTOR commands must be issued one-at-a-time and the host must wait for a response. That response can either be the return some requested disk data, status data, or an error code if MSTOR detects a mal-formed command. The commands which can return lots of block-framed data (Read/Write/Append/List) require a command sequence consisting of the initial command which returns or accepts the first data block, then a signal from the data receiver to either send the next data block, repeat the last block, or abort the command. The commands which send or retrieve small amounts of non-disk data to/from MSTOR are not sent in CRC'd block frames. Any non-CRC'd settings or readings that the host considers critical (perhaps date/time setting, or communications interface selection, etc) can be read multiple times for verification if needed.

All MSTOR commands start with the '&' character followed by one command letter. These 2-byte commands are intentionally short, without CRC or other embedded error checking. This makes it easy to manually send or monitor these commands during development/debugging. The command length is embedded in the command so that MSTOR knows when each command is complete. Most responses consist of the '#' character followed by the command byte , then two decimal digits representing response length, then any response data. The exceptions are the commands which transfer repeated blocks of disk data such as when reading/writing files or listing files/directories.

TABLE 1 above holds the list of available commands. These can be sent to MSTOR through any interface selected .. the command set for all interfaces is identical. Commands are described as characters sent serially, however they can also be considered as bytes sent via I2C. When the specified number of command characters has been received, the command is assumed complete and is checked for format and executed. Every command results in a response even if it is just an acknowledge.

MSTOR includes CRCs in all disk-read data blocks. The host can validate them by re-computing its own CRCs and comparing the two, or it can ignore the CRCs received. And when the host writes data to disk, MSTOR will validate the received CRC if the block is terminated with '>', or ignore the CRC if the received block is terminated with '@', so CRC usage is completely controlled by the host.

The RS232 interface is always 'active' even if another interface is selected as the communications interface. However once another interface is selected, the command set which the RS232 interface accepts is limited to (I)nterface selection/identification, and (C) Save Configuration Parameters . That means that it (RS232 port) will not accept disk commands or other MSTOR configuration commands. All commands will be accepted and acted upon through the currently selected interface, and the RS232 port can only identify and/or change the selected interface, and save parameters. MSTOR is delivered with the RS232 interface selected. Using the provided PC companion program, the user can

set/verify the timestamp for the local time zone, select appropriate command timeouts, select encryption, read/write a test file to verify USB disk formatting/compatibility etc… all while the RS232 interface is the selected interface and the user has the human visibility offered by a PC monitor. Then before deploying the system a command can be sent to change the main interface (to I2C or RS485) if desired. Again, if the communications interface is set to one other than RS232, the RS232 port will only accept the Interface selection/identification commands – all other commands will be ignored..

In the command details below characters enclosed in [] brackets represent one or more binary bytes. For instance [ACK] represents the single ASCII ACK character (0x06), [CRC] represents two binary bytes

Command Details

****

LIST FILES and/or DIRECTORIES Commands … list files or subdirectories or both from the current directory. All files are listed even if they have system, or hidden attributes. The encryption setting does not apply to this command. File and directory names are not encrypted. The syntax of the command is

1. **&L05**F .. &L with the F parameter is the list-files command, 05 is the 2-digit total number of characters in the command. MSTOR returns the names, date/timestamps, and sizes of all the files in the current directory one at a time. Each list item frame starts with a block start character '<' and ends with block terminator '>'. One binary length byte [L] and two CRC bytes [CRC] are included. FILESIZE is a string of decimal digits representing the total size of the file in bytes. A single SPACE character (0x20) follows each element of a file item.. one after each of the filename, date, time, and filesize. A typical block frame is shown below.

   <[L]FILENAME.TXT 12/04/2018 11:06AM FILESIZE [CRC]>

   Note that the length of this returned block exceeds 32 characters, so if the block size selected is 32, all list items will be broken into two blocks,, the first containing the filename and date (and a CRC for the block), and the second block will contain the time and file size and a CRC.

   The host must issue a single 'N' character to retrieve the next item,/block, 'Y' to repeat the last block or 'X' to exit and abort this command. The host continues to issue 'N' characters until MSTOR returns the empty block **<[0x05][CRC]>** indicating that the command is complete and MSTOR is ready for a new command. If the host aborts (using 'X' response) MSTOR does NOT send the empty block as above it just aborts the command and is ready for a new one. To prevent unending loops, MSTOR allows a maximum of 5 consecutive retries of the same block before

issuing an error and aborting the listing. And if a read timeout occurs because the host did not respond with an N or Y or X within the *read timeout* period set, MSTOR returns an error code "#L11Error-08" and aborts the command.

a. **&LxxFmask** (xx > 5 and xx < 18) can be issued instead which includes a file mask . Same as list command above except that the return is restricted to those file entries matching the mask. Use the * character as a wildcard match to multiple characters, and the '?' as a single wild character match…. *.*, F??.*, FILE.TXT, G*.T?? are all examples of valid masks.

2. **&L05D** .. &L with the D parameter is the list-directories command, 05 is the 2-digit total number of characters in the command. This command returns the names of the sub-directories in the current directory. Following each directory name is "$D" which indicates that the name represents a directory name. Each list item frame starts with '<' and ends with block terminator '>'. A one byte message length and two byte CRC are included. The host must issue an 'N' character to retrieve the next item, or 'X' to abort. When the host receives an empty block (as above) the command is complete and MSTOR is ready for a new command. 32 byte block sizes may cause each directory item to be split into two framed blocks as described above.

   a. **&LxxDmask** (xx > 5 and xx < 18) can be issued instead which includes a directory mask . Same as list command above except that the return is restricted to those sub directory entries matching the mask. Use the * character as a wildcard match to multiple characters, and the '?' as a single wild character match…. *.*, D33??.*, DIR.TXT, K2*.T?? are all examples of valid masks.

3. **&L05A** ..&L with the A parameter is the list-all command, 05 is the 2-digit total number of characters in the command. This command returns both (sub) directories and files in the current directory. Each directory item is followed by the "$D" indicator. Each list item frame starts with '<' and ends with block terminator '>'. A one binary byte message length and CRC are included. The host must issue an 'N' character to retrieve the next item, or 'X' to abort the command.
   Note that this command returns data in a format that is similar to the way a Windows Explorer tree-display returns data. That is, only the current branch data is returned (just sub directories and files of the current directory), and directory items are grouped ahead of file items. Each element of the item is followed by a SPACE character (0x20) then the last modified date, then another SPACE, then the last modified time, a SPACE, the file size, a SPACE, a possible $D directory indicator, a SPACE, a CRC and block terminator. Each item is returned in its own data block, and after each block MSTOR waits for the host to send an 'N' (send next item) or X (exit-abort listing). &L05A (list All sub directories and files) data would look like the following if the data block size is large enough to contain an entire data block.

<[L]DATADIR 12/02/2018 11:06AM 0 $D [CRC]>
…MSTOR waits for N, then sends
<[L]NEXTDIR 04/15/2016 02:22PM 0 $D [CRC]>
... MSTOR waits for N, then sends
<[L]FILE1.ABC 07/09/2002 07:08PM 25345 [CRC]>
... MSTOR waits for N, then sends
<[0x05][CRC]>

The last empty block indicates that there are no more items to list, and the command is complete. . If the data block size selected is 32 bytes then blocks will be split into two framed blocks with the first block containing the item name and date (and a CRC). The 2nd block contains the timestamp and file size and possibly the $D directory marker and a CRC. These two blocks representing a single split-block item would look like:

<[L]DATADIR 04/08/2012 [CRC]>
... MSTOR waits for N, then sends
<11:06AM 0 $D [CRC]>

The host repeatedly sends the 'N' sub-command to get the next block of file/directory data until it receives an empty block which marks the end of the listing. Instead of sending an 'N', the host can send an 'X' which will cause MSTOR to abort the listing by sending the empty block **<[0x05][CRC]>** and terminating the command.

****

CHANGE DIRECTORY Command … change directory up or down in the tree from the current directory. No CRC is sent and the command is not block framed. The syntax of the command is :

1. **&DxxDIRNAME** .. &D is the change directory command, xx is the 2 decimal digit total number of characters in the command – in this case xx = 11 . This command moves down into a sub-directory. If the operation is a success, the return is "#D05[ACK]" and if the specified directory is not found an error code ("#D12Error-22" for instance) is returned. DIRNAME can specify multiple levels of directory movement such as topdir/mydir/newdir/ in a single command however the length of DIRNAME must be 28 characters or less.

2. **&D06..** &D is the change directory command, 06 is the 2-digit total number of characters/bytes in the command, and the two periods in place of a name tells MSTOR to move up one directory towards the root of the drive. (Just like Windows, UNIX, and DOS a single period refers to the current directory, and two periods refers to the one above the current directory). If successful this command

returns a "#D05[ACK]". If there was an error it returns "#D12Error-xx" where xx is a two decimal digit error code.

3. **&D04** This command returns the full path/name of the current directory from the root directory in the standard format **\dir1\nextdir\nextlevel\currentdir .** Since the returned string can be very long, this command returns a block-formatted result. The current block size determines the length of these blocks which are formatted identically to all 'L' and 'R' command data. The host must send a single 'N' character to retrieve the (N)ext string or 'Y' to repeat a block, or 'X' to abort the command. The maximum path length returned is 128 characters. Each framed block is CRC'd.

****

MAKE DIRECTORY Command … Create a new sub directory in the current directory. The syntax of the command is :

&MxxSUBDNAME &M is the make new directory command followed by xx which are the two decimal digits noting the length of this command (in this case 12) followed by up to 12 characters in 8.3 format for subdirectory name. Note that the subdirectory must not already exist in the current directory . If successful this command returns "#M05[ACK]. If there was an error it returns "#M12Error-xx" where xx is a two decimal digit error code.

****

KILL(DELETE) DIRECTORY Command … Delete a subdirectory from the current directory. The syntax of the command is :

&Kxx[MODE]SUBDNAME &K is the delete directory command followed by xx which are the two decimal digits noting the length of this command (in this case 13). [MODE] is one of two characters indicating the mode of operation :

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.

SUBDNAME is up to 12 characters for subdirectory name in 8.3 format. Note that the subdirectory must exist in the current directory or the command returns an error. (The root directory cannot be deleted. If successful this command returns "#K05[ACK]. If there was an error it returns "#K12Error-xx" where xx is a two decimal digit error code.

****

RENAME FILE Command … Rename a file in the current directory. The syntax of the command is :

&NxxFILENAME.EXT>NEWNAME.TXT &N is the rename command followed by xx which are the two decimal digits noting the length of this command (in this case 28) followed by up to 12 characters for current file name, the '>' character, followed by up to 12 characters of the new filename. If the file does not exist in the current directory MSTOR will return an error. If successful this command returns "#N05[ACK]". If there was an error it returns "#N12Error-xx" where xx is a two decimal digit error code. Wildcard characters are not allowed since only one specific file can be renamed.

****

ERASE (Delete) FILE Command … Delete the specified file from the current directory. The syntax of the command is :

&ExxFILENAME.EXT &E is the delete file command followed by xx which are the two decimal digits noting the length of this command (in this case 16). The file must reside in the current directory. If found, it is deleted and if not there, an error is returned. No confirmation questions are asked … like 'Are you sure you want to delete?' .. the file is simply deleted. The response for a successful operation is "#E05[ACK]" or an error message if a problem was detected.

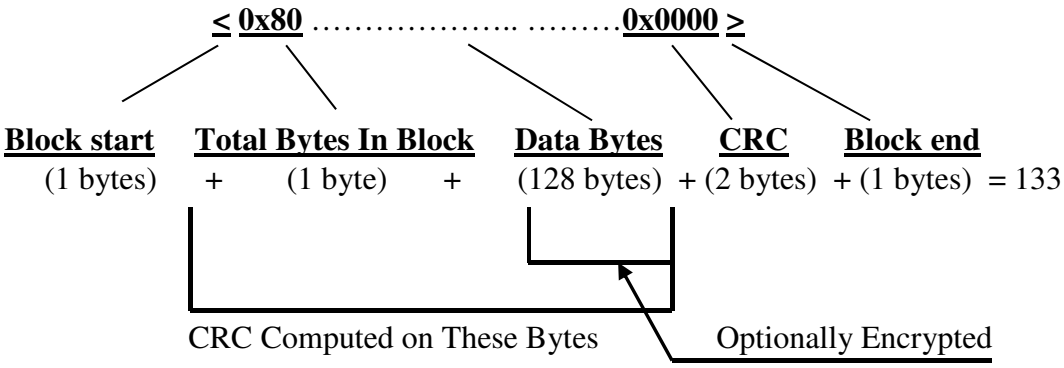
****

READ FILE Command … Read the specified file from beginning to end. The syntax of the command is :

&RxxFILENAME.EXT &R is the read file command followed by xx which are the two decimal digits noting the length of this command (in this case 16). The file is returned in 32/64/128 byte data blocks depending on the selected block size, and output to the specified interface. Each block of data starts with the framing character '<' followed by *one byte* (not decimal digits) indicating the total count of response bytes, then the number of data bytes defined by the block size setting (32/64/138/256), then a 2-byte CCITT CRC16 computed for the data bytes and the one byte count of data bytes , and terminated with the framing character '>'. So the user typically receives a total of 1 + 1 + 128 + 2 + 1 = 133 bytes for each block for a block size of 128.

It is recommended that the user validate each block CRC, and to accommodate even the slowest, lower power host systems, MSTOR does not output the next data block until the host returns a single **'N'** character (read-Next block) sub-command, or a **'Y'** sub-command character which tells MSTOR to retry the last block (used if there is a CRC failure), or a single **'X'** sub-command which causes MSTOR to Abort the active Read command and close the file. The host usually issues repeated 'N' sub-commands to

MSTOR until it detects an empty block indicating that MSTOR has closed the file after reaching the end. MSTOR is then ready to accept a new command. The CRC is returned in big-endian format (MSbyte first). A typical 128 byte data block looks like this:



The block end character can alternatively be '@' which only the host can send. This informs MSTOR that the CRC bytes are NOT to be validated. The host may stuff dummy values in its blocks if the repeated CRC block calculations would occupy too much of its CPU/MCU processing time.
As described above, MSTOR sends each block then waits for an N or Y or X before sending another block or aborting, however it does not wait indefinitely. MSTOR provides a recovery mechanism in case the host fails to issue one of these sub command characters. The user-programmable *read timeout* period is in effect during file reads. If MSTOR starts this read sequence, and the host unexpectedly stops requesting blocks (before the end-of-data is reached and MSTOR has returned a empty block) for longer than this period, MSTOR issues its own abort and returns an error message, closes the open file, and is ready to accept a new command. There is no ACK response as there is with other commands.

****

WRITE FILE Command … Write the specified file into the current directory. The syntax of the command is :

&WxxFILENAME.EXT &W is the write file command followed by xx which are the two decimal digits noting the length of this command (in this case 16). If there is an existing file of the same name, it will be overwritten ***with no user notification***. The host must wait until MSTOR is ready to start accepting data blocks for this file. If there is no file of that name a new file will be created. After the file is created and ready for writing, MSTOR issues the single 'N' character telling the host to send the (N)ext block. The host must frame the data starting with '<' followed by one byte indicating the total bytes in the frame, then up to 128 bytes of file data for a 128-byte block size, a 2-byte big endian CRC16 and ending with '>'or alternatively '@'. Although use of the CRC is recommended, the host can bypass its use by assigning the two CRC16 bytes any value, and terminating the block with '@' . MSTOR will not attempt to validate the CRC for blocks terminated with this terminator. If there is any error in the command (too many data bytes, invalid format, invalid CRC etc) instead of returning a request for the next block of

data, MSTOR will return 'Y' which is the request for a retry of the block. The maximum number of consecutive retries for any block is 4. If after 4 attempts the received block cannot be validate, MSTOR issues an error message and aborts the command. The host can abort the command at any time by sending a single unframed 'X' character instead of a framed block of data. When MSTOR detects this it will immediately close the current file and terminate the Write command. Data must be framed in blocks as in the R command above, and the CRC usage is optional although recommended. To end the Write command the host sends an empty block **"<[0x05][CRC]>"** . MSTOR closes the file, returns "#W05[ACK]" and terminates the command.
If MSTOR must terminate the command for any reason (disk full, disk was removed, file was unable to be overwritten due to write protections etc) instead of asking for the next block with an 'N' response, it will issue "#W12Error-xx' which is the error indicator. The xx represents two decimal digits identifying the error code. MSTOR closes the file if possible and aborts the command.
.

****

APPEND FILE Command … Append data to the specified file in the current directory. The syntax of the command is :

&AxxFILENAME.EXT &A is the append file command followed by xx which are the two decimal digits noting the length of this command (in this case 16). If there is an existing file of that name, data will be appended to it. The host must wait until MSTOR is ready to start accepting data blocks to add to this file. After the file is opened and ready for writing, MSTOR issues the 'N' character telling the host to send the (N)ext block. If there is no file found for appending, MSTOR will return '#A12Error-xx" which is the error indicator. No file will be created and the command will be aborted.
The remainder of the operation is identical to the W command above except that MSTOR responds with "#A05[ACK]" after the last empty data block.

****

SEEK FILE Command … Advance (SEEK) a specified number of bytes from the start of the file and begin reading from that point. There are no relative seeks allowed (advancing from a previous seek position) – all seeks are performed on a file that is not already open, and the seek advances from the beginning of the file.

&FxxyyyyyyyyyyFILENAME.EXT &F is the SEEK command followed by xx which are the two decimal digits noting the length of this command (in this case 26) . yyyyyyyyyy represents exactly 10 decimal digits representing the byte-count to advance from the beginning of the file. The maximum byte advance allowed is 4294967295. This command operates identically to the Read File command above except that it starts reading from the byte-position. If no matching file is found, or if the starting position specified is greater than the length of the file, an error is returned. Wild-card characters are not allowed. MSTOR returns the data in block format with each block CRC'd, and the host

must respond to each block with an 'N' to get the (N)ext block, (Y) to retry/resend the previous block, or 'X' to abort the command. The host continues to read blocks until MSTOR reaches the end of the file and returns the empty block .. **"<[0x05][CRC]>"** . Please refer to the Read File command above.

****

GET FILE SIZE Command … Return the size of the specified file in bytes. The syntax of the command is :

&SxxFILENAME.EXT .. &S is the get file size command followed the two decimal digit command length. The file must reside in the current directory. If found, MSTOR will return the total number of bytes in the file as a string of decimal digits up to a maximum of 4294967295 . This is the maximum number that can be represented by a (32-bit) unsigned long integer. Example: "#S0944683 . #S is the command response identifier, 09 is the total length of the response and 44683 is the length of the file.

****

INTERFACE SELECT Command … Select the interface for disk communications. The syntax of the command is :

1. **&IxxTTT[Addr]** &I is the interface select command, xx is the two decimal digits noting the length of this command -- always 07 or 08), and TTT are the 3 characters that identify the type of interface RS232, RS485, or I2C to be used for the transfer of disk data. [Addr] represents one additional binary byte (0x00 – 0x7F) defining the I2C address which is accepted only if TTT = "I2C".

TTT = "232" identifies the RS232 interface
TTT = "485" identifies the RS485 interface
TTT = "I2C" identifies the I2C interface with a one-byte I2C slave address appended

Note that the RS232 and RS485 interfaces can be active simultaneously … that is, the RS485 interface can be selected above as the disk data transfer interface while the RS232 interface is still available to set the real time clock, or set a different disk interface. Disk related commands through the RS232 interface are disabled when another interface has been chosen as the disk data transfer interface. MSTOR returns "#I05[ACK]"

2. **&I04** This version of the command returns the identity of the currently selected interface – it returns the same 3 or 4 characters that were used to set the interface…. 232, 485, or I2C[Addr] ([Addr] is one byte of data 0x0-0x7F indicating the currently set 7-bit I2C slave address). This command can be issued through the currently selected interface, or through the RS232 interface. The response is sent to the issuing interface: **#IxxTTT[Addr]** with the last [Addr] byte only included if TTT = "I2C" as above.

****

SET DATA BLOCK SIZE Command .. Set the maximum number of data bytes contained in any block-framed message to 32/64/128/256 . Disk file data is broken into blocks for transmission to/from the host. This command allows a host to size the data blocks so its buffers will not be overrun. The total block lengths including the '<' block start marker, the byte count of the block, the payload bytes, the two-byte CRC and the '>' block end marker are 37/69/133/261 bytes respectively.

1. **&ZxxBLKSIZE** &Z is the block size command followed by xx which is the 2-decimal digit length of this command (06 or 07). BLKSIZE is the 2 or 3 decimal digit value defining the maximum block size. Valid values for BLKSIZE are 32, 64 and 128.
   MSTOR will format all disk data in blocks with this as their maximum size, and the host must adhere to this length as well. Since data blocks including < …> framing, their data payloads are less than the block size and are shown below.
   &Z0632
   &Z0664
   &Z07128
   &Z07256

   MSTOR responds with #Z05[ACK]

2. **&Z04** &Z is the block size command followed by 04 which is the 2-decimal digit length of the command. This command allows the host to query MSTOR for the current block size. MSTOR returns #ZxxBLKSIZE where xx and BLKSIZE are the same as above.

****

SET R/W TIMEOUT Commands … Set the timeouts to be used during block transfers. The WriteTimeoutValue is in effect when the host is writing to MSTOR, and the ReadTimeoutValue is in effect when the host is reading from MSTOR. These are the maximum time periods that MSTOR will wait between < …> data blocks before timing out and aborting the current transfer command. They provide an automatic recovery mechanism if the repeated handshake exchange of data blocks unexpectedly ends. Timeout values are specified in 10ms increments from 1 (10ms) to 3000 (30 seconds). These values can be saved in nonvolatile storage so do not need to be reset each time MSTOR is powered.

1. **&H09W0025** .. &H is the set timeout command, 09 is the two decimal digits noting the length of this command, followed by the W character which means that MSTOR is to set the *WriteTimeoutValue* to the value contained in the next 4 decimal digits. In this case the write timeout is being set to 25 X 10ms = 250

milliseconds. The factory-set default value is 100ms. MSTOR responds with "#H05[ACK]"

2. **&H09R0100** .. &H is the set timeout command, 09 is the two decimal digits noting the length of this command, followed by the R character which means that MSTOR must set the *ReadTimeoutValue* to the value contained in the next 4 decimal digits. In this case the read timeout is being set to 100 X 10ms = 1 second. The factory-set default value is 100ms. MSTOR responds with "#H05[ACK]"

****

SET RS485 Baud Rate Commands … Set the baud rate for the RS485 port. RS485 is a differential signaling system which offers more reliability at higher baud rates than RS232. Therefore this port can run faster than the RS232 port. &B is the command followed by two decimal digits of command length. The available options are:

1. **&B089600** set to 9600 baud
2. **&B0919200** set to 19.2k baud
3. **&B0938400** set to 38.4k baud
4. **&B0957600** set to 57.6k baud
5. **&B10115200** set to 115.2k baud
6. **&B10200000** set to 200kbaud
7. **&B10250000** set to 250k baud

MSTOR responds with "#B05[ACK]"

****

TURN ENCRYPTION ON/OFF Command

1. **&J05N** Turn disk encryption ON. MSTOR responds with "#J06N[ACK]" . Files read are automatically decrypted with the currently set encryption key. Files written are encrypted with the currently set encryption key. It is the host's responsibility to maintain key/file encryption integrity -- so that MSTOR has the correct encryption key for each encrypted file. Appending to an encrypted file directly is not possible since it must be a continuous process involving the entire file. Appending requires reading/decrypting the existing file, appending to the unencrypted version, then re-encrypting the entire file.

2. **&J05F** Turn disk encryption OFF. MSTOR responds with "#J06F[ACK]"

3. **&J04** Commands MSTOR to return the current encryption setting as #J05N for ON, or #J05F for OFF.

****

SAVE Configuration Values Command

4. **&C05** Saves the currently set MSTOR configuration parameters to non-volatile memory. This command should be issued after initial configuration, and after any of the single-asterisk commands (See Table 1 above) are executed to change a parameter. MSTOR's non-volatile memory can be re-written a minimum of 10,000 times but the host should make all parameter changes, then issue a single SAVE command to store them all at once and conserve the available write cycles. These non-volatile values are restored whenever MSTOR powers up or is reset so that prior operation can resume without any reconfiguration. This insures that if MSTOR undergoes a power failure/power return that the host is unaware of, it will continue to operate as it did prior to the power failure. MSTOR responds with "#C05[ACK]" when complete. Typical execution time is 50 ms.

****

SET TIME Command … MSTOR contains a real-time clock so that files can be timestamped when written/changed. It also contains a battery which keeps this clock running when MSTOR is not powered. If an accurate/updated timeclock is available to the host system, then periodically the MSTOR clock should be synchronized using this command. MSTOR's internal clock is accurate to a few seconds a month so synchronization is typically necessary only yearly. No block formatting is used and there is no CRC included. The syntax of the command is :

&T20YYYYMMDDHHMMSSxM &T is the set time command followed by two decimal digits representing the length of the command, then year, month , day, hour ,minute and second. x - represents A or P so that the last two characters identify AM or PM. Time is always represented in a 12-hour format. No block formatting is used and no CRC is included. MSTOR returns the acknowledge string "#T05[ACK]" or an error string. The setting can be verified by reading the time back with the GET TIME command.

****

GET TIME Command … MSTOR can serve as the system time-reference by providing its real-time clock value to its host. Or this command can be used as a confirmation that the time setting was received correctly. The syntax of the command is :

&G04 &G is the get time command followed the two decimal digit command length. It is the request for MSTOR to return its date/time. MSTOR responds a string formatted like the one used to set the time. That is: "**#G20YYYYMMDDHHMMSSxM"**

With the x representing A or P so that the last two characters identify AM or PM. No block formatting is used and no CRC is included.

****

Set ENCRYPTION KEY Command

1. **&Y20KEY …**&Y is the set encryption key command and xx is the two decimal digits indicating the number of characters in this command . KEY is a 16 character string which is your own key to your unique encryption. (KEY must not contain a '&' character since MSTOR interprets that as the start of a new command.) MSTOR does NOT retain this key in non-volatile memory so it is not available for use after power-up like other configuration settings. It can be read back to validate its setting and the host should verify it before it is used to encrypt files. A corrupted setting is like an unknown/lost key. The host must retain its key(s) so it can decrypt files previously encrypted. If you set the key = "0000000000000000", MSTOR will use its own default (hidden and unreadable) 128 bit key which is unique to each MSTOR device. Therefore any encrypted data written with one MSTOR device using its default key is NOT readable with a different MSTOR device using its default key. The MSTOR default key is not alterable by the user, and always reads back as "0000000000000000" although that is not the actual key. Any files encrypted/stored using the default key will be readable (by the same MSTOR only) by setting the encryption key back to "0000000000000000" which restores MSTOR to its unique internal key which is its power-up setting. Using the MSTOR default key does not protect disk data if both MSTOR and the thumb drive are lost/stolen as a pair, but it does protect disk data if the disk is separated from its paired MSTOR and is lost or stolen. The response to this command is "#Y05[ACK]" . Example command:

   &Y20HERE IS MY KEYAA

2. **&Y04** .. this command returns the current key. Before reading or writing encrypted files the host should use this command to verify that the key setting is as expected. MSTOR may have experienced a power failure and recovery that the host is unaware of in which case MSTOR has powered up with its default key. So encryption operations might proceed with an unexpected key. By validating the key both before and after any encrypt/decrypt operations the host can be assured that the operation occurred with the desired key and that a brown-out did not occur causing the key to revert back to the MSTOR default. If the key value AFTER an operation is not as expected, the host can 'un-do' the operation by reversing the most recent operation (leaving the current key in place), then change the key and 're-do' the operation with the desired key.

   MSTOR responds with #Y20KEY where KEY is as above.

****

POWER ON/OFF Command … Power Up/Down the USB drive. For extended periods of non-use the drive can be powered down to save system power. NOTE: MSTOR is designed to operate with small self-contained thumb drives and this assumes a small solid state low power disk that derives its power from MSTOR's USB connector. It is possible to plug in a rotating-disk USB drive cable which has its own external power source. In this case this command will not be able to control disk power. The syntax of the command is :

1. **&P05N** &P is the power control command followed by 05 which are the two decimal digits noting the length of this command followed by the N character commanding MSTOR to turn the drive power ON. The response is "#P05[ACK]" . When MSTOR powers on it automatically turns disk power ON so this command is not needed unless the host has since turned disk power OFF.

2. **&P05F** &P is the power control command followed by 05 which are the two decimal digits noting the length of this command followed by the N character commanding MSTOR to turn the drive power OFF. The response is "#P05[ACK]"

3. **&P04** is the power control command which allows the host to determine the power status/install state of the USB drive. This command returns "#N04[ACK] if the drive is powered ON, and has been successfully enumerated (connected to) the USB interface, so is ready to use. If power is not applied or the drive did not successfully enumerate this command returns "#F04[ACK]" which means power is OFF, or a drive is not installed, or the drive failed to enumerate.
   The USB enumeration process can take several seconds so if the drive power is turned ON using the command above, the host should wait before checking the drive's status (or issuing any drive commands). USB drives respond differently so there is no absolute time limit on readiness after the application of power. The operator also has a visual indication of drive readiness since MSTOR's green flashing LED flashes very quickly (about 5 times per second) before enumeration, and much more slowly (about once per second) after enumeration.

4. **&P09SLEEP** is the special power command which puts MSTOR into deep sleep after responding with "#P05[ACK]". In this SLEEP state with 5 volt still applied, the 5V input current drops to less than 40 microamps . If 5V power is then removed, the internal battery takes over to power the real time clock requiring just 1-2 microamps from the internal battery. It is not necessary to issue this command before removing 5V power .. MSTOR detects that condition and puts itself to sleep. And when 5V power is reapplied, it auto-resets itself and returns to operation.
   However if 5V power is removed abruptly during a disk transfer (power failure or disk unplugged with an open file) this is similar to a PC's 'unsafe disk removal' condition, and the integrity of the open file and filesystem is not guaranteed.

For 6-12VDC operation, MSTOR's sleep current with power applied is approximately 10ma due to internal regulator operation, and when this power is removed, again the internal battery takes over as above.

****

ENCRYPT EXISTING FILE Command … Create an encrypted disk file from an unencrypted disk file.

&QxxUNENCR.EXT>ENCR.EXT &Q is the command UNENCR.EXT is the name of an existing unencrypted file in the current directory, and ENCR.EXT is the name of the encrypted file you would like to create. ENCR.EXT must not already exist and both filenames must conform to 8.3 format. The command ignores the Encryption ON/OFF setting (J command above) and will encrypt the specified file using the currently defined key -- either your currently assigned key, or the default MSTOR key (see the Y command above). No further user intervention is required, and MSTOR will respond with #Q05[ACK] when done – or #Q12Error-xx if an error occurred due to disk full, file not found etc. The original unencrypted file is NOT automatically deleted. Note that a previously encrypted file can be encrypted again with the same or a different key, however to recover the unencrypted data the file has to be decrypted twice using the last-used-key first, then again with the first-used-key. The host must manage all encryption key/file associations.

****

DECRYPT EXISTING FILE Command … Create an encrypted disk file from an unencrypted disk file.

&UxxENCR.EXT>UNENCR.EXT &U is the command ENCR.EXT is the name of an existing encrypted file in the current directory, and UNENCR.EXT is the name of the unencrypted file you would like to create. UNENCR.EXT must not already exist and both filenames must conform to 8.3 format. The command ignores the Encryption ON/OFF setting (J command above) and will decrypt the specified file using the currently defined key -- either your currently assigned key, or the default MSTOR key (see the Y command above). No further user intervention is required, and MSTOR will respond with #Q05[ACK] when done – or #Q12Error-xx if an error occurred due to disk full, file not found etc. The original encrypted file is NOT automatically deleted. Note that a previously unencrypted file can be decrypted however the resulting data will be unintelligible. If this is done inadvertently, the data can be recovered by un-doing that operation with a re-encryption (using the same key) to restore the original file.

****

GET CODE VERSION Command … Return the code version that MSTOR is currently operating. The syntax of the command is :

&V04 &V is the version command followed the two decimal digit command length. It is the request for MSTOR to return its current operating version string. Example:

#V05MSTOR v1.04

****

ABORT COMMAND

'&' This special command character can be issued at any time during the current command sequence. It indicates the start-of-command and appears nowhere else in any command sequence. Therefore anytime MSTOR receives this character when awaiting a command (outside of a data block transfer) it assumes that this is the start of a new command . (This character must NOT be included in an encryption key string). When detected, MSTOR aborts receipt of any partially received command sequence and flushes its command buffer. It can be used to resynchronize the command processor with the host if the host does not get a response when it expected one (possibly due to too few characters in a previous command, or other command corruption condition, or power-cycling of MSTOR). Note that it does not abort any of the Read/Write/Append/Seek file commands which are already in process. Those sequential command processes can be aborted by the host sending an 'X' instead of an 'N' as a block retrieval notice (refer to the above command descriptions for details). Any of the sequential read or write commands can also be aborted simply by waiting for the appropriate *timeout value* previously set to expire without communication.

A hardware /RESET signal is also provided and can be issued to restart and reinitialize MSTOR to its power-up state. Notice that this should not be activated while reading/writing to disk or disk corruption could occur. (This would be similar to an 'unsafe disk removal' on a PC).

Encryption and Security:

MSTOR provides 128-bit AES encrypt/decrypt features to protect the data stored on disk. You can define your own 128-bit (16 character) encryption key, or use the default MSTOR key and both schemes have security implications:

- **a)** If the USB drive *only* is lost or stolen either key provides security of your data since the data would be VERY difficult to access without the key.
- **b)** If both MSTOR and the drive are stolen, and you had defined your own encryption key – which MSTOR does NOT save – your data is still secure. The key is NOT stored in nonvolatile memory. However if you used the default MSTOR key for encryption, that key IS available and can be used to decrypt your data by anyone who has the *paired* MSTOR and thumb drive. (Each MSTOR has its own unique default, unreadable key).

So it can be better to create your own key and send it to MSTOR, however there can still be issues if you use serial communications between MSTOR and your host and the 'wire' is accessible to data interception. The key itself is vulnerable and if intercepted, would compromise the security of the encrypted data on disk. In this case, a better encryption scheme is to perform encryption/decryption with your own system (MSTOR's encryption turned off) . No encryption key would be transmitted, and only encrypted data would pass across the wire so sniffing is defeated. This does make it more difficult to transfer the drive to a PC and transfer data, however the MSTOR Configuration application provided by Versalent allows you to read/write from an encrypted USB drive by entering your key.

There are various combinations of events and connections which could affect security differently and it is the user's responsibility to assess the vulnerability and implement the system accordingly.

Timekeeping and File TimeStamps:

When files are written/modified it is important to store/update the file timestamp, and this timestamp should be agree closely with the current date-time standard in your zone. Because many small microcontroller systems do not provide real-time clocks to maintain this date-time, nor a means of accessing an internet time-server, MSTOR provides a battery-backed real-time clock which keeps accurate time even during very long power outages. This allows file timestamps to be updated accurately as files are written/appended even without access to a time server.

During manufacture and testing each MSTOR's internal clock is set to Pacific Standard Time. Prior to installation the operator can use the MSTOR Configuration Application on a PC to alter this time as necessary to agree with local time and/or periodically correct for

MSTOR ‘clock-drift’ . MSTOR’s clock is accurate to a few seconds per month and its internal battery will keep this clock active for more than 5 years in the power-down state. The CR2025 non-rechargeable battery is replaceable by opening the case and should be replaced every 5 years if MSTOR is powered down for long periods of time. After battery replacement, MSTOR’s time will need to be reset. MSTOR time is readable by the host so it can act as the real-time reference for your system.

Reliability Features:

MSTOR implements an internal watchdog timer, and an internal brown-out detector. If the internal microcontroller gets disrupted through static discharge or other temporary interference, the watchdog will automatically reset the unit so that normal operation resumes with no user intervention. The USB drive does not have to be reseated ... MSTOR will power-up and re-initialized the USB connection and begin operating. If the supplied power droops below an operational threshold the brown-out detector will suspend operation until power is restored to a normal level. At that point it will resume operation from a reset state again generally with no user intervention. If input power droops low enough, MSTOR automatically enters deep sleep mode – everything shuts down except the real-time clock which is powered by the battery. When power returns (and transitions to an operational level in 200ms or less) MSTOR will automatically reset and resume operation from its deep-sleep mode. If power rises very slowly, the auto-reset is not guaranteed and MSTOR may remain in deep sleep requiring a 10 usec minimum low pulse on the /RESET input to resume disk operations.

RS232 Baud Rate Control:

The RS232 baud rate is set using the 2 internal shorting jumpers as shown below. To access these jumpers the case must be opened.

MSTOR’s RS232 baud rate can be set to any of the values shown in TABLE 1. To change it the 2 internal shunt blocks are moved to the positions shown. Very slow baud rates are not supported since they are not useful for file transfer. Also parity is not supported since the block-CRC that MSTOR provides is significantly better at detecting multiple bit errors. The table shows shaded blocks where shunts are to be installed to select the RS232 baud rate. After changing these settings MSTOR must be powered off/on to reset and accept the new serial settings.

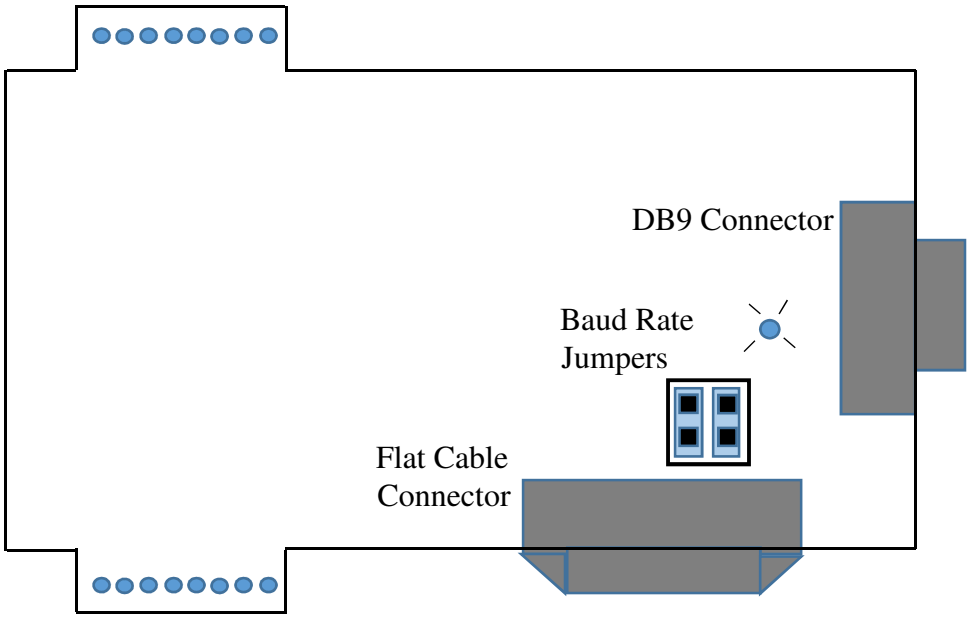


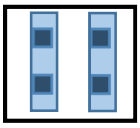
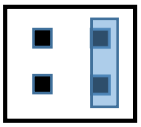
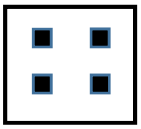
DIAGRAM	RS232 BAUD RATE
	9600 bps
	19.2k bps
	57.6k bps
	115k bps

TABLE 1

.

MSTOR devices are configured for 115k 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. This is because the internal Texas Instruments MAX202 RS-232 driver has a receiver threshold that is approximately +1.2 volts above ground. So any 'single-ended' signals which do not go below ground provide a sufficient RS-232 signal level for the device to operate reliably. Please refer to:

http://www.ti.com/lit/gpn/max202 for more technical details of the driver's capabilities.

RS485 Baud Rate Control

The baud rate for the RS485 port is separate from the RS232 port speed (it is implemented using a separate UART). Since the RS485 signals are differential, the speed can be higher than RS232 while maintaining data integrity. The differential paid of RS485 signals is terminated with 300 ohm inside MSTOR – this is a typical impedance of a twisted pair cable and/or a pair of adjacent wires of a ribbon cable. This is only approximate but it provides some amount of matching of the differential transmission line and provides better signal integrity than unterminated wires. The RS485 baud rate is set with the /B command above.

Power Control:

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 (or 6-12VDC for –V models) to a pin of the 20-pin ribbon connector
2) Plug a 6-12VDC power adapter/brick into the MSTOR power jack

Applying Power Through the Ribbon Connector:

Power can be applied to PIN 3 of the ribbon connector**. Depending on the model ordered, this can be 5VDC @ 100ma or 6-12 VDC @ 100ma for models with an internal regulator. The 6-12V option is useful for remote operation (RS232 or RS485 at a distance) where a long cable would drop more than 0.15V @ 100ma. Or a remote MSTOR could be powered from a 6-12V wall adapter with a 2.1mm X 5mm center positive barrel jack. The internal regulator in MSTOR will then supply the needed +5VDC.

**Note that RS232 handshake pins cannot supply sufficient power for a USB drive and MSTOR – external power is required.

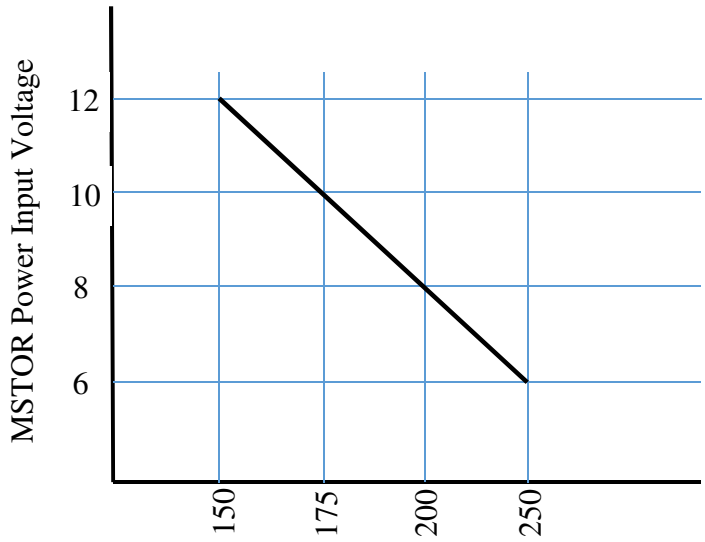
MSTOR MODELS AVAILABLE

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 50-100ma, and while standard USB ports can provide up to 500ma for device operation or charging, MSTOR is limited to USB drives that require no more than 250ma. See following power-derating curve.



MSTOR POWER DERATING @25C

In addition to the input voltage derating, temperature derating must be applied. For each 10°C rise in ambient temperature above 25°C , the above maximum current should be derated an additional 20ma.

MSTOR Physical, Electrical, Environmental Specifications

Physical:

Size:	(3.8" x 2.4" x 0.9"
Weight:	3.3 oz
USB Connector:	Standard Type-A
Host Connector:	Standard 20-pin ribbon cable and/or screw terminals
Case Color:	Almond
Power Connector:	5mm X 2.1 mm (Ctr Positive)

Electrical:

Absolute Maximum Input Voltage	+15VDC (Internal Regulator Models only)*
Nominal Voltage Input:	+5VDC +/- 5% or 6-12VDC (per model number)
Current Input:	25mA plus USB drive current
Baud Rate (RS232 port):	9600 to 115k selectable
Baud Rate (RS485 port)	Programmable 9600 to 250k
Digital I/O Signal Level	3.3V logic compatible.
RS232 Signal Inputs	+/- 25V absolute maximum HIGH threshold 2.4V minimum LOW threshold 0.8V maximum
RS485 Signal Inputs	+/- 14V absolute maximum Minimum differential threshold +/- 0.2V

Environmental:

Max Operating Temperature:	+75ºC
Min Operating Temperature:	10ºC
Max Storage Temperature:	+90ºC
Min Storage Temperature:	-20ºC
Humidity:	Non-condensing at all temperatures

Document Revision Record:

Revision #	Revision Date	Description
V--	Oct, 2018	Preliminary document