SCSI Toolbox, LLC
PO Box 620520
Littleton, CO 80162-0520
ph: 303.972.2072
fax: 720.981.7737
www. SCSItoolbox.com

SCSI Toolbox, LLC
Developer Toolbox
VC SCSI Extension Functions – July 17, 2003

Creating a VCSCSI DTB Project

Step 1: Copy the file ***VCPSSLImports.h*** into the project folder

Step 2: In your StdAfx.h file, add the line
#include "VCPSSLImports.h

Step 3: Insure that STBVCBase.dll, STBVCBase.lib, VCPSSL.dll, and VCPSSL.lib are in the System32 folder, or you can copy the above 4 files into your project folder

Step 4: -
- Select the Visual Studio Main Menu **'Project'** choice
- Select the **'Settings'** choice
- Highlight your project name in the left pane, then click the **'Link'** tab
- Choose the **'General'** selection from the **'Category'** combo selection box
- In the **'Object/Library Modules'** edit box type ***"VCPSSL.LIB"***
-

IMPORTANT: When debugging, you'll need to tell VC++ the relative path to VCPSSL.lib by going to the Link tab, insure the "Category" combobox has "Input" selected, and then in the "Additional library path" editbox, enter the relative path. In most cases, you should enter "..\" without the quotation marks.

Developer Toolbox VCSCSI Extension Functions

MultiThreaded SCSI Commands

VCSCSIIssueThreadedCDB

(int nHA,int nTid,int nLun,BYTE * pCDBBytes,int nCDBLen,BYTE * pInOutBuf,int nInOutBufLen,int nDirection,int nTimeout) As Integer

Issues a User-defined MultiThreaded CDB to the specified ha/target/lun address.

The CDB is defined as follows:

- The CDB is defined in the array pCDBBytes
- The CDB size (6,10,12, or 16 bytes) is defined in nCDBLen
- The data buffer used by the cdb is defined by the byte array pInOutBuffer
- The data buffer length is defined by the long nInOutBufferLen
- The data direction (1 = in from target, 0 = out to target) is defined by nDirection
- The CDB timeout (in seconds) is defined by nTimeout

Returns: Zero on failure, or a ThreadID on success

VCSCSIGetThreadedCDBStatus

(int nThrdID) As Integer

Checks the status of the CDB associated with the Thread ID *nThrdID*

Returns:

one of the following:

0 = eTestInProgress

1 = eCompleteOnSuccess

2 = eCompleteOnFailure

3 = eTestNotStartedYet

4 = eTestIsPaused

5 = eTestStopped

6 = eErrorOnParamsPassed

7 = eMisCompare

8 = eInvalidBlock

9 = ePendingIOOutstanding

10 = eUnknownStatus

VCSCSIGetThreadedCDBStatusWData

int nThrdID,BYTE * pSenseBuf,int nSenseBufLen,double * pTimeToDoCmd,BYTE * pInOutBuf,int nInOutBufLen) As Integer

Use this function to check the thread status and to retrieve performance metrics and data if your CDB had a data in phase.

The array pSenseBytes – array length specified by nSenseBufLen – will return sense data if there is any from a CDB that caused a check condition.

pTimeToDoCmd will return the time that it took the CDB to execute

pInOutBuffer will contain nInOutBufferLen bytes of data if your CDB had a data-in phase

Returns:

one of the following on failure:

0 = eTestInProgress

1 = eCompleteOnSuccess

2 = eCompleteOnFailure

3 = eTestNotStartedYet

4 = eTestIsPaused

5 = eTestStopped

6 = eErrorOnParamsPassed

7 = eMisCompare

8 = eInvalidBlock

9 = ePendingIOOutstanding

10 = eUnknownStatus

VCSCSIReleaseThreadID

(int nThrdID) As Boolean

It is very important to call this function, passing it the ThreadID, when your command is complete.

Returns:

1 on success, non-1 on failure

VCSCSIAnd

(int number1, int number2) As Integer

Logically AND's number1 with number2

Returns: results of AND operation

VCSCSIBuffer2File

(int buffnum, int datalen, char *FileName) As Integer

Writes *datalength* bytes of buffer # *buffnum* into file *filename.*
The file filename is opened for append, and is created if it does not exist.

Returns: returns 0 on success, non-zero on failure

VCSCSIBufferSize

() as long

Returns the maximum size of buffer 0.

VCSCSICheckRandomBlock

(int buffer, int blocksize, long blocknum, long offset, int numberofblocks)As Integer

Checks the data in ***buffer*** to make sure that the block numbers and random data are correct. ***Blocksize*** specifies the size of the expected data blocks, ***blocknum*** specifies the first block number in the series of blocks, ***numberofblocks*** specifies how many blocks are expected to be in the buffer. ***Offset*** allows you to index into the buffer to a specific block.

Return value is –1 if the check is successful. If the return value does not equal –1 you should call VCSCSIGetRandomErrors to determine the cause of failure.

VCSCSICompareBuffers

(long startbyte, long numbytes) As Integer

Compares the contents of two buffers buffer 0 and buffer 1), starting from *startbyte*, for length *numbyte*.

Returns: returns 0 on success, byte number of miscompare on failure

VCSCSICMQ

As Integer

Clears the Windows message queue so messages from button pushes, etc, can be processed. Call this function before checking status.

Returns: returns 0 on success, non-zero on failure

VCSCSIDec2Hex

(long ul, BYTE *hexdata) as integer

This function accepts a decimal number, then converts it into hexadecimal and fills in a array of four bytes with the hex values.

Returns: returns 0 on success, non-zero on failure

VCSCSIDiskCorruptBlock

(int ha, int target, int lun, long sblock, int span) as integer

This function corrupts the block specified by blocknum, on the drive specified by ha/target/lun, with an error that is eccspan bits long.

Return value is the status of the WRITE LONG cdb – 0 = command complete (success), 2 = check condition (failed)

VCSCSIDiskGetECCSpan

(int ha, int target, int lun) as long

This function issues a MODE SENSE to read Mode Page 1 from the disk specified by ha/target/lun.

The return value is the Correction Span value reported in the Error Correction mode page.

VCSCSIGetReadLongSize

(int ha, int target, int lun) as long

Returns the number of bytes that should be used as the blocksize parameter in the VCSCSIDiskReadLong and VCSCSIDiskWriteLong functions.

VCSCSIDiskRead

(int ha, int target, int lun, int count, long sblock, long bsize, int buffer) As Integer

Reads *count blocksize* size blocks of data, starting at *highblock*, from drive specified by *ha/target/lun* into *buffer*.

Returns: returns 0 on success, non-zero on failure

VCSCSIDiskReadFUA

(int ha, int target, int lun, int count, long sblock, long bsize, int buffer) As Integer

This function is the same as VCSCSIDiskRead, but it sets the Force Unit Access bit of the CDB.

See documentation on VCSCSIDiskRead

VCSCSIDiskReadLong

(int ha, int target, int lun, int correct, long sblock, int bsize, int buffer) As Integer

Issues a VCSCSI READ LONG cdb to read ***blocksize*** bytes from block ***highblock*** of the drive specified by ***ha/target/lun***. The correct bit of the cdb can be set with the ***correct*** parameter. The data is read into the buffer specified by ***buffer***.

Use the VCSCSIGetReadLongSize function to determine the correct blocksize to specify.

Return value will be 0 for COMMAND COMPLETE, 2 for CHECK CONDITION

VCSCSIDiskStartStop

(int ha, int target, int lun, int start) As Integer

Starts (*start* = 1) or stops (*start* = 0) drive specified by *ha/target/lun.*

Returns: returns 0 on success, non-zero on failure

VCSCSIDiskUnload

(int ha, int target, int lun) As Integer

Ejects/Unloads media from drive specified by *ha/target/lun.*

Returns: returns 0 on success, non-zero on failure

VCSCSIDiskVerify

(ha as long, target as long, lun as long, count As Integer, highblock As Long, blocksize As Long, buffer As Integer) As Integer

Issues a VCSCSI VERIFY command to the drive specified by *ha/target/lun.* Verifies *count blocksize* blocks of data, starting at *highblock*.

Returns: returns 0 on success, non-zero on failure

VCSCSIDiskWrite

(int ha, int target, int lun, int count, long sblock, long bsize, int buffer) As Integer

Writes *count blocksize* size blocks of data, starting at *highblock*, to drive specified by *ha/target/lun* into *buffer*.

Returns: returns 0 on success, non-zero on failure

VCSCSIDiskWriteFUA

(int ha, int target, int lun, int count, long sblock, long bsize, int buffer) As Integer

This function is the same as VCSCSIDiskWrite, but it sets the Force Unit Access bit of the CDB.

See documentation on VCSCSIDiskWrite

VCSCSIDiskWriteLong

(int ha, int target, int lun, long sblock, int bsize, int buffer) As Integer

Issues a VCSCSI WRITE LONG cdb to write ***blocksize*** bytes from block ***highblock*** of the drive specified by ***ha/target/lun***. The data is written from the buffer specified by ***buffer***.

Use the VCSCSIGetReadLongSize function to determine the correct blocksize to specify.

Return value will be 0 for COMMAND COMPLETE, 2 for CHECK CONDITION

VCSCSIDLT_FWDL

(int ha, int target, int lun,char *FileName) as Integer

Download firmware to a Quantum DLT drive

Returns: 0 on success, -2 if firmware file not found, -1 if download fails

VCSCSIFile2Buffer

(int buffnum, int datalen, char *FileName) As Integer

copies *datalength* bytes from file *filename* into buffer # *buffnum* .

Returns: returns 0 on success, non-zero on failure

VCSCSIFileOffset2Buffer

(int buffnum, int datalen,char *FileName, long offset)
As Integer

copies *datalength* bytes from file *filename* into buffer # *buffnum* .
The data is copied from *offset* bytes from the beginning of the file.

This function is useful if you need to copy the contents of a file to a VCSCSI device in "chunks", for example, use this function to download firmware to a device using segmented WRITE BUFFER commands.

Returns: returns 0 on success, non-zero on failure

VCSCSIFillBlockNum

(int buffer, long sblock, int count, int blocksize) As Integer

Fills buffer # *buffer* with disk *blocknumber* data, starting at byte 0 of buffer, continuing for *count* X *blocksize* bytes (*count* blocks of data)

Returns: returns 1 on success, -1 on failure

VCSCSIFillBuffer

(int buffer, long numbytes, int patternsize,BYTE *bufferdata) as Integer

Fills buffer # *buffer* with *count* bytes of data. The buffer is filled with the pattern specified in the *bufferdata()* . The number of bytes in the data pattern is specified by *pattsize*.

Example:

For example, you may specify a data pattern that is:

- 4 bytes long
- consists of the hex data 01, 02, 03, 04

Then you could repeat this pattern into the first 1,024 bytes of buffer 0

With the command:
Ret = VCSCSIFillBuffer(0,1024,4,databuf)

Returns: returns 1 on success, -1 on failure

VCSCSIFillPattern

(int buffer, int pattern) as Integer

Fills buffer # *buffer* with one of the following patterns, based on *pattern*:

0 – all zeros – 0
1 – all ones – 1
2 – alternating 0/1 – 0xA5
3 – alternating 1/0 – 0x5A
4 – random data

Returns: returns 1 on success, -1 on failure

VCSCSIFillRandom

(int buffer, int blocksize, long startingblock, int numberofblocks) As Integer

Fills the specified buffer with numberofblocks blocksize blocks of random data. The first four bytes of each block of data contain the block number, the next four bytes contain the seed used to generate the random data for that block.

Return value is the number of blocks generated.

VCSCSIGetBuffer

(int buffer, long numbytes, BYTE *bufferdata) as Integer

Retrieves *count* bytes of data from buffer # *buffer*, returns data in byte array *bufferdata*.

Returns: returns 1 on success, -1 on failure

VCSCSIGetBufferMode

(int ha , int target , int lun) as Integer

Retrieves the current buffer setting of the selected tape drive.

Returns: returns the current buffer mode setting

VCSCSIGetDeviceType

(int ha, int target, int lun) as Integer

Issues a VCSCSI INQUIRY command to drive specified by *ha/target/lun* and returns PERIPHERAL DEVICE TYPE value (INQUIRY byte 0).

Returns: PERIPHERAL DEVICE TYPE on success, -1 on failure.

VCSCSIGetDllVersion

() as Integer

Returns the current version of the PTI pssl dll

VCSCSIErrorDetails

(int *iostat, int *hbastat, int *scsistat) as integer

Retrieves the SRB (Driver), HBA, and SCSI Target status from the most recently issued CDB.

Returns 0

VCSCSIGetProduct

(int ha, int target, int lun, char *ProductString) as Integer

Issues a VCSCSI INQUIRY command to drive specified by *ha/target/lun* and returns PRODUCT ID value (INQUIRY bytes 16-31) in string *product*.

Returns: returns 1 on success, 0 on failure

VCSCSIGetRandomErrors

(long *expected_blocknum, long *actual_blocknum, int *block, int *offset, int *expected_data, int *actual_data) As Integer

Returns error information describing a VCSCSICheckRandomBlock failure.

Return value will be 1 if valid error information is available, -1 if the error information is invalid. The values returned in the passed parameters will be –1 if they are invalid, otherwise they will contain information that will show if the block number did not compare (expected_blocknum is what the block number should have been, actual_blocknum will be what blocknumber was read). If a data byte does not compare block will contain the block number the error occurred in, offset will contain the byte offset within that block, and expected_data and actual_data will show the data error.

VCSCSIGetTapeCapacity

(int ha, int target, int lun, long *TBS) as Integer

Issues a VCSCSI MODE SENSE command to drive specified by *ha/target/lun*, returns Tape Block Length (Mode Page Block Descriptor bytes 5-7) in long tbs.

Returns: returns 1 on success, -1 on failure

VCSCSIGetVendor

(int ha, int target, int lun, char *VendorString)) as Integer

Issues a VCSCSI INQUIRY command to drive specified by *ha/target/lun* and returns VENDOR ID value (INQUIRY bytes 8-15) in string *vendor*.

Returns: returns 1 on success, 0 on failure

VCSCSIGetVersion

(int ha, int target, int lun, char *VersionString)) as Integer

Issues a VCSCSI INQUIRY command to drive specified by *ha/target/lun* and returns VERSION ID value (INQUIRY bytes 32-35) in string *version*.

Returns: returns 1 on success, 0 on failure

VCSCSIHex2Dec

BYTE *hexdata) as long

This function accepts an array of four bytes, then converts it into decimal number and returns that number

Returns: decimal number

VCSCSIHostAdapterCount

() as Integer

Returns: number of VCSCSI host adapters in system on success

VCSCSIHPLTO_FWDL

(ha as long, target as long, lun as long, FileName as String) as Integer

Download firmware to a HP LTO drive

Returns: 0 on success, -2 if firmware file not found, -1 if download fails

VCSCSIIBMLTO_FWDL

(int ha, int target, int lun,char *FileName) as Integer

Download firmware to a IBM LTO drive

Returns: 0 on success, -2 if firmware file not found, -1 if download fails

VCSCSIInitializeElementStatus

(int ha, int target,int lun)as Integer

Issues an Initialize Element Status command to the addressed jukebox device.

Returns: Zero on success, non-zero on failure

VCSCSIInitializeElementStatusRange

(int ha, int target,int lun, int range,int address, int num)as Integer

Issues an Initialize Element Status with Range command to the addressed jukebox device.

The Range field indicates which elements to initialize. If range = 0 initialize all elements. If range = 1 initialize the range of elements specified by the address and number fields.

Returns: Zero on success, non-zero on failure

VCSCSIInquiry

(int ha, int target, int lun, BYTE *inqdata) as Integer

Issues a VCSCSI INQUIRY command to drive specified by *ha/target/lun* and returns "raw" INQUIRY data (INQUIRY bytes 0 –63) in byte array inqdata.

Returns: returns 1 on success, 0 on failure

VCSCSILoadBuffer

int buffer, long numbytes, BYTE *bufferdata) as Integer

Fills buffer # *buffer* with *count* bytes of *bufferdata*.

Returns: returns 1 on success, -1 on failure

VCSCSILogSense

(int ha, int target, int lun, int page, int pagecode, BYTE *logdata) as Integer

Issues a LOG SENSE command for log page *page*, page code *pagecode* to the drive specified by *ha/target/lun*. Log Sense data is returned in the byte array *logdata*.

Returns: returns 1 on success, -1 on failure

VCSCSIModeSense

(int ha, int target, int lun, int page, int pagecode, BYTE *modedata) as Integer

Issues a MODE SENSE command for mode page *page*, page code *pagecode* to the drive specified by *ha/target/lun*. Mode Sense data is returned in the byte array *modedata*.

Returns: returns 1 on success, -1 on failure

VCSCSIModeSelect

(int ha, int target, int lun, int sp, BYTE *modedata) as Integer

Issues a MODE SELECT command to the drive specified by *ha/target/lun*. MODE PAGE data is transferred from byte array *modedata*, excluding BLOCK DESCRIPTOR DATA.

Returns: returns 1 on success, -1 on failure

VCSCSIModeSelectFull

(int ha, int target, int lun, int sp, BYTE *modedata) as Integer

Issues a MODE SELECT command to the drive specified by *ha/target/lun*. MODE PAGE data is transferred from byte array *modedata*, including header + block descriptor + page data.

Returns: returns 1 on success, -1 on failure

VCSCSIModeSenseFull

(int ha, int target, int lun, int page, int pagecode, BYTE *modedata) as Integer

Issues a MODE SENSE command for mode page *page*, page code *pagecode* to the drive specified by *ha/target/lun*. MODE SENSE command is issued with DBD (Disable Block Descriptor) bit NOT set, therefore block descriptor data IS transferred. Mode Sense data (header + block descriptor + page data)is returned in the byte array *modedata*.

Returns: returns 1 on success, -1 on failure

VCSCSIMoveMedium

(int ha, int target, int lun, int trans_add, int source_add, int dest_add)as Integer

Issues an Move Medium command to the addressed jukebox device.

Transport = media transport (picker) address

Source = source element address
Destination = destination element address

Returns: Zero on success, non-zero on failure

VCSCSIOr

(int number1, int number2) As Integer

Logically OR's number1 with number2

Returns: results of OR operation

VCSCSIPositionToElement

(int ha, int target, int lun, int trans_add, int dest_add)as Integer

Issues a Position to Element command to the addressed jukebox device.

Transport = media transport (picker) address
Destination = destination element address

Returns: Zero on success, non-zero on failure

VCSCSIReadCapacity

(int ha, int target, int lun, long highblock_l, long blocksize_l) as Integer

Issues a VCSCSI READCAPACITY command to drive specified by *ha/target/lun* and returns the devices HIGHBLOCK # in *highblock* and BLOCKSIZE in *blocksize*.

Returns: returns 0 on success, non-zero on failure

VCSCSIReadElementStatus

(int ha, int target,int lun, int eltype, int start, int num, long length, BYTE *eldata) as Integer

Issues a Read Element Status command to the addressed jukebox device.

Elementtype = jukebox element type requested – 0 for all types
Startelement = elements equal to or greater than the starting address are returned
Num = number of element descriptors to return
Length = byte length allocated for returned element descriptors
Eldata = pointer to an array of bytes to hold returned element descriptors

Returns: Zero on success, non-zero on failure

VCSCSIResetHBA

(int hba) as Integer

This function causes a bus reset and a bus rescan of the specified HBA

Returns: returns 0 on success, non-zero on failure

VCSCSIRollPattern

(int buffer, long start, long number) as Integer

Writes *number* bytes of data into the buffer specified by *buffnum*. The data written begins with the 4-byte long value specified by *start*. This value is incremented by one each time it is written into the buffer.

Returns: returns 0 on success, non-zero on failure

VCSCSISEAGATELTO_FWDL

(int ha, int target, int lun,char *FileName) as Integer

Download firmware to a Seagate LTO drive

Returns: 0 on success, -2 if firmware file not found, -1 if download fails

VCSCSISonyAIT_FWDL

(int ha, int target, int lun,char *FileName) as Integer

Download firmware to a Sony AIT drive

Returns: 0 on success, -2 if firmware file not found, -1 if download fails

VCSCSISearchBuffer

(int buffer, BYTE *searchdata, int searchsize, long searchlength) as Integer

Searches buffer # *buffer* for the first occurance of *searchdata*. *Searchlength* specifies how much of the buffer to search (-1 searches the entire buffer), *searchsize* specifies the number of significant bytes in the pattern *searchdata*.

Returns: returns byte count of the first byte of buffer that matches pattern on success, -1 on failure

VCSCSISDLT_FWDL

(int ha, int target, int lun,char *FileName) as Integer

Download firmware to a Quantum SuperDLT drive

Returns: 0 on success, -2 if firmware file not found, -1 if download fails

VCSCSISegmented_FWDL

(int ha, int target, int lun,char *FileName) as Integer

Download firmware to a device

Returns: 0 on success, -2 if firmware file not found, -1 if download fails

VCSCSISetTimeout

(int seconds) as Integer

Sets the CDB timeout value in seconds. This timeout value will remain in effect until the next VCSCSISetTimeout function is called.

Specifying a value (seconds) greater than zero will set the timeout to that value in seconds. Specifying a value of zero will set the default timeout to 30 seconds. Specifying a value of less than zero will set the timeout to infinite.

Returns: 1 on success, -1 on failure

VCSCSITapeBlockSize

(int ha, int target, int lun, long blocksi) as Integer

Sets the blocksize of drive specified by *ha/target/lun* to *tbs*.

Returns: 1 on success, -1 on failure

VCSCSISetBufferMode

(int ha , int target , int lun , int buffermode) as Integer

Sets the buffer mode of the selected tape drive

Returns: zero on success, non-zero on failure

VCSCSITapeRewind

(int ha, int target, int lun, int immediate) as Integer

Rewinds the drive specified by *ha/target/lun*. Returns after rewind completion if *immediate* = 0, or upon acceptance of cdb by device if *immediate* = 1.

Returns: 1 on success, -1 on failure

VCSCSITapeUnload

(int ha, int target, int lun, int immediate) as Integer

Unloads the drive specified by ha/target/lun. Returns after unload completion if immediate = 0, or upon acceptance of cdb by device if immediate = 1.

Returns: 1 on success, -1 on failure

VCSCSITapeWFM

(int ha, int target, int lun) as Integer

Writes a FILE MARK to drive specified by ha/target/lun.

Returns: 1 on success, -1 on failure

VCSCSITapeReadF

(int ha, int target, int lun, int count, int buffer) as Integer

Reads *count* fixed blocks from the tape specified by *ha/target/lun* into buffer # *buffer*.

Returns: 1 on success, -1 on failure

VCSCSITapeWriteF

(int ha, int target, int lun, int count, int buffer) as Integer

Writes *count* fixed blocks to the tape specified by *ha/target/lun* from buffer # *buffer*.

Returns: 1 on success, -1 on failure

VCSCSITapeReadV

(int ha, int target, int lun, int buffer) as Integer

Reads one variable block from the tape specified by *ha/target/lun* into buffer # *buffer*.

Returns: 1 on success, -1 on failure

VCSCSITapeWriteV

(int ha, int target, int lun, long count, int buffer) as Integer

Writes one variable block *count* bytes to the tape specified by *ha/target/lun* from buffer # *buffer*.

Returns: 1 on success, -1 on failure

VCSCSITapeFSF

(int ha, int target, int lun) as Integer

Spaces forward on file mark on the tape specified by *ha/target/lun*

Returns: 1 on success, -1 on failure

VCSCSITapeFSR

(int ha, int target, int lun) as Integer

Spaces reverse on file mark on the tape specified by *ha/target/lun*

Returns: 1 on success, -1 on failure

VCSCSITapeSpaceEOD

(int ha, int target, int lun) as Integer

Spaces forward to END OF DATA on the tape specified by *ha/target/lun*

Returns: 1 on success, -1 on failure

VCSCSITargetCount

(int ha) as Integer

Returns: number of targets supported on host adapter *ha*

VCSCSITUR

(int ha, int target, int lun) as Integer

Issues a TEST UNIT READY command to the drive specified by *ha/target/lun*.

Returns: returns 0 on success, non-zero on failure

VCSCSIUserCdb

(int ha, int target, int lun, BYTE *cdb, int cdblength, int datadir, long datalength, int buffer, int tOut) as Integer

Issues the VCSCSI CDB secified in byte array *cdb* to the device specified in *ha/target/lun*. The length of the CDB is specified in *cdblength*, data direction is specified by *datadir*(0=out from host adapter, 1 = in to host adapter), length of data transferred is specified by *datalength*, and buffer # is specified by *buffer*.

Returns: 1 on success, -1 on failure

VCSCSIViewSense

(BYTE *reqsendata) as Integer

Returns the latest REQUEST SENSE data in byte array *sensedata*.

Returns: 1 on success, -1 on failure

VCSCSIXor

(int number1, int number2) As Integer

Logically XOR's number1 with number2

Returns: results of XOR operation