Windows DLL Functions for Developers of TimeWand I and DuraWand Applications (Windows 95/98/NT)

GCO# 1120

Introduction:

Videx provides software tools that enable developers and programmers to seamlessly integrate TimeWand I/DuraWand communications capabilities with applications software. One of these tools is **DLOAD32.DLL**.

DLOAD32.DLL is a 32-bit library of functions that enable Windows applications to retrieve data from the TimeWand I or DuraWand. These functions are compatible with Windows development tools that are capable of calling DLLs. The **DLOAD32.DLL** functions are documented below.

In order to use **DLOAD32.DLL**, **the developer must understand DLLs and how to call DLL functions**. Videx provides documentation on the **DLOAD32.DLL** functions only. The developer must know how to call DLL functions from his/her own Windows development environment.

This document contains definitions for each **DLOAD32.DLL** function or subroutine, illustrating the function return type and parameter types with **long** to indicate a numeric (long) integer, and **str** to indicate a string. All of the **DLOAD32.DLL** functions return a long integer value.

Three of the functions contain parameters for technical support. The **traceDebug** parameter requires knowledge of the actual software and can be useful if you are calling for technical support. The **IOdebug** parameter will show, in the same file as the **traceDebug** parameter, all of the data and commands sent to and received from the TimeWand I/DuraWand. This debug data will be written to a file named **DEBUG.TXT**.

There will also be a file written named **STATUS.TXT** that gives the status of each operation. This file can be overwritten or appended. If a wand did not communicate with the computer, its wand ID will not appear in the status file.

As an example, sample declarations and function calls for “Visual Basic” are included at the end of this document.

DLL Function Calls:

long InitializeWand (long WandID, str symPath, str aplPath, long ComPortNumber, long initTime, long traceDebug, long IOdebug)

Initializes a TimeWand I or DuraWand by loading the wand with the specified symbology program file, wand ID, and application file. There must be only one wand in the downloader during initialization. The wand must have been reset (triple beep reset).

Parameter	Description
WandID	Long integer containing the ID that is to be assigned to the TW1/DW. The value must be a valid long integer less than 999,999,999.
symPath	String containing the symbology program filename, including the full path (for example: **c:\videx\3of9kp.twp**).
aplPath	String containing the application filename, including the full path (for example **c:\videx\app.txt**. If no application is to be loaded in the wand, this parameter is an empty string. Refer to page 7 for information on the contents of this file if loading an application (hierarchy or sequence).
ComPortNumber	Long integer containing the number (1–9) of the desired communication port.
InitTime	Long integer containing the time allowed (in seconds) for the communications software to receive a response from the TW1/DW. If a time of zero (0) seconds is used, the DLL will default to 64 seconds. Once the TW1/DW is placed in the recharger/downloader, it wakes up once each minute, so a value of at least 60 seconds is recommended. The maximum allowed value of an integer is system dependent, but large values serve no real purpose for this function.
traceDebug	Long integer to set a debug flag that will create a file named **DEBUG.TXT**. This file contains traces of the procedure names that have been called in the DLL. A value of one (1) turns **traceDebug** on; a value of zero (0) turns **traceDebug** off. This is helpful for troubleshooting problems with the help of Videx Customer Support.
IOdebug	Long integer to set a debug flag that will create a file named **DEBUG.TXT**. This file contains a copy of all of the communications the DLL has had with the TW1/DW, from the perspective of the DLL. A value of one (1) turns **IOdebug** on; a value of zero (0) turns **IOdebug** off. This is helpful for troubleshooting problems with the help of Videx Customer Support.

Returns: If the initialization is successful, **InitializeWand** returns zero (0). If there are any errors, **InitializeWand** returns a negative number. The return values are listed on page 6.

long DownLoadWand (long WandID, str DataFileName, long ComPortNumber,
long OverWriteData, long MemoryDump, long ClearMemory,
long DLtime, long traceDebug, long IOdebug)

Transfers data from a wand and appends or overwrites the data to an ASCII text data file (raw scan file format). The wand ID which is used must match the ID with which the wand was initialized.

Parameter	**Description**
WandID	Long integer containing the ID that was assigned to this wand. The value must be a valid long integer less than 999,999,999.
DataFileName	String containing the name of the ASCII text file in which to store the data from the wand. If the file does not exist, it will be created. If the file already exists, the data will be appended to the end of the file or the file will be overwritten (depending upon the value of the **AppendOrWrite** flag).
ComPortNumber	Long integer containing the number (1–9) of the desired communications port.
OverWriteData	Long integer to set a flag to determine whether the output file will be appended or overwritten. With a value of zero (0), the output data file will be appended or created. With a value of one (1), the output data file will be overwritten or created.
MemoryDump	Long integer to set a flag to determine whether to download scanned data or to dump the wand's memory. With a value of zero (0), the wand will have its scanned data downloaded to the designated file. With a value of one (1), the wand will have its entire data area downloaded to a file named **MEMORY.DMP**.
ClearMemory	Long integer to set a flag to determine whether to clear the wand's memory after a successful download. With a value of one (1), the wand's memory will be cleared after the download is complete. With a value of zero (0), the DLL will not clear the memory of the wand. The wand will not be reset with this selection, and may start "chirping" soon after the DLL completes communications. The user must press the scan button to stop the "chirping."
DLtime	Long integer containing the time allowed (in seconds) for the DLL to hear back from the requested wand. Once the wand is placed in the recharger/downloader, it will wake up once each minute, so a value of at least 60 seconds is recommended. The maximum allowed value of an integer is system dependent, but large values serve no real purpose for this function.
traceDebug	Long integer to set a debug flag that will create a file **DEBUG.TXT**. This file contains traces of the procedure names that have been called in the DLL. A value of one (1) turns **traceDebug** on; a value of zero (0) turns **traceDebug** off. This is helpful for troubleshooting problems with the help of Videx Customer Support.
IOdebug	Long integer to set a debug flag that will create a file **DEBUG.TXT**. This file contains a copy of all of the communications the DLL has had with the TW1/DW, from the perspective of the DLL. A value of one (1) turns **IOdebug** on; a value of zero (0) turns **IOdebug** off. This is helpful for troubleshooting problems with the help of Videx Customer Support.

Returns: If the download is successful, **DownLoadWand** returns zero (0). If there are any errors, **DownLoadWand** returns a negative number. The return values are listed on page 6.

long DownLoadWandList (str WandIDlist, str DataFileName, long ComPortNumber,
long OverWriteData, long ClearMemory, long DLtime,
long traceDebug, long IOdebug)

Transfers the data from wands whose IDs are specified in a list. Appends or overwrites the data to an ASCII text data file (raw scan file format).

Parameter	**Description**
WandIDlist	String containing the name of the ASCII text file containing the wand IDs to be transferred. The file format is: <wandID><cr/lf>.
DataFileName	String containing the name of the ASCII text file in which to store the data from the wands. If the file does not exist, it will be created. If the file already exists, the data will be appended to the end of the file or the file will be overwritten (depending upon the value of the **AppendOrWrite** flag).
ComPortNumber	Long integer containing the number (1–9) of the desired communications port.
OverWriteData	Long integer to set a flag to determine whether the output file will be appended or overwritten. With a value of zero (0), the output data file will be appended or created. With a value of one (1), the output data file will be overwritten or created.
ClearMemory	Long integer to set a flag to determine whether to clear the wand’s memory after a successful download. With a value of one (1), the wand’s memory will be cleared after the download is complete. With a value of zero (0), the DLL will not clear the memory of the wand. The wand will not be reset with this selection, and may start “chirping” soon after the DLL completes communications. The user must press the scan button to stop the “chirping.”
DLtime	Long integer containing the time allowed (in seconds) for the DLL to hear back from the requested wands. Once the wand is placed in the recharger/downloader, it will wake up once each minute, so a value of at least 60 seconds is recommended. The maximum allowed value of an integer is system dependent, but large values serve no real purpose for this function. Each time a wand responds, the **DLtime** value is reset to the original value sent to the DLL. Therefore each wand is given the same amount of time in which to be downloaded.
traceDebug	Long integer to set a debug flag that will create a file **DEBUG.TXT**. This file contains traces of the procedure names that have been called in the DLL. A value of one (1) turns **traceDebug** on; a value of zero (0) turns **traceDebug** off. This is helpful for troubleshooting problems with the help of Videx Customer Support.
IOdebug	Long integer to set a debug flag that will create a file **DEBUG.TXT**. This file contains a copy of all of the communications the DLL has had with the TW1/DW, from the perspective of the DLL. A value of one (1) turns **IOdebug** on; a value of zero (0) turns **IOdebug** off. This is helpful for troubleshooting problems with the help of Videx Customer Support.

Returns: If the download is successful, **DownLoadWandList** returns zero (0). If there are any errors, **DownLoadWandList** returns a negative number. The return values are listed on page 6.

long SerOpen (long ComPortNumber)

Opens and initializes the specified serial port on the PC. The **SerOpen** function is called automatically by **InitializeWand**, **DownloadWand**, and **DownloadWandList** and should not be called in conjunction with these functions. It is provided here only as a utility.

Parameter	**Description**
ComPortNumber	Long integer containing the number (1–9) of the desired communications port.

Returns: If the serial port is opened and initialized successfully, **SerOpen** returns zero (0). If any errors are encountered, **SerOpen** returns a negative number. The return values are listed on page 6.

long SerClose (long ComPortNumber)

Closes the specified serial port on the PC. The **SerClose** function is called automatically by **InitializeWand**, **DownloadWand**, and **DownloadWandList** and should not be called in conjunction with these functions. It is provided here only as a utility.

Parameter	**Description**
ComPortNumber	Long integer containing the number (1–9) of the desired communications port.

Returns: If the serial port is closed successfully, **SerClose** returns zero (0). If any errors are encountered, **SerClose** returns a negative number. The return values are listed on page 6.

long SetStatusMode (long statusMode)

Sets the options for the status file to report the result of communication.

Parameter	**Description**
statusMode	Flag to set status file options. If 0, then there is no status file (though the DLL will still create a zero length file). If 1, the status information will be appended to an existing status file. If 2, the existing status file will be overwritten with the status information from this communication.

Returns: **SetStausMode** returns **statusMode**. There are no error codes that are returned by this function.

Sub SetProgress (long userFunction)

Sets up a callback function to enable user cancellation of communication. The user function declaration should be:

Long Function userFunction (long dummy1, long dummy2)

The **dummy1** and **dummy2** parameters will always have a 0 value in this implementation. Future versions of this software may use these parameters to allow implementation of customized progress functions.

Parameter	**Description**
userFunction	Long integer containing the address of the user function that checks for cancellation of communication.

Returns: Nothing.

Return Values:

The **DLOAD32.DLL** functions return long integer values. If no error occurs, the functions return 0. Return values from -1 to -12 are Windows errors. Return values from -13 to -52 are DLL errors. The errors are listed in the following table.

0	noErr	No error
-1	IE_BADID	The device identifier is invalid or unsupported. (Does not know about the port ID that it was given.)
-2	IE_OPEN	The device is already open. (Port is already open.)
-3	IE_NOPEN	The device is not open. (Port is not open.)
-4	IE_MEMORY	The function cannot allocate the queues. (Not enough memory.)
-5	IE_DEFAULT	The default parameters are in error.
-10	IE_HARDWARE	The hardware is not available. (Locked by another device.)
-11	IE_BYTESIZE	The specified byte size is invalid.
-12	IE_BAUDRATE	The device's baud rate is unsupported.
-13	badParamErr	Incorrect parameter in function call.
-14	portOpenAlready	Serial port already open.
-15	otherPortOpen	Other serial port is open.
-16	openButBadBaud	Serial port open but not initialized properly.
-18	portNotOpen	Serial port not open.
-20	alreadyRunning	**DLOAD32** is already being run by another application.
-21	hexToASCIIErr	Error converting hex data to ASCII.
-22	noInputFile	Error accessing file for reading.
-23	noOutputFile	Error opening data file.
-24	commErr	Communications error or serial port problem.
-25	timeoutErr	Wand stopped responding to the computer.
-28	cksumErr	Communications error, data block checksum error.
-30	wandNotFound	Wand did not respond.
-36	tooManyRetries	Communications problem.
-37	statusFileErr	Error writing to **STATUS.TXT** file.
-38	inputFileErr	Error reading wand list for **DownLoadWandList**.
-39	noWandId	No valid wand ID was specified in the function call.
-43	invalidVersion	Wand has invalid version to complete communications.
-44	wandNotCleared	Wand has valid data (found during initialization).
-50	quitErr	User pressed Alt-F4 to exit the application or Windows.
-51	abortErr	User pressed Esc or right mouse button to abort.
-52	mismatchedFiles	The symbology selected does not match the symbology file name in the application file.

Application File Format:

The format of the application file expected by the **InitializeWand** function is similar to that expected by the DOS command line utility **dload.exe**. This is done for compatibility reasons. This file consists of text lines of parameters separated by spaces and terminated with a carriage return and line feed. The menu-driven communications software (**TW1.EXE** or **DW.EXE**) creates this file automatically and calls it **TW1LIST.TXT**.

The application file has the format:

<wand ID><reloadFlag><initialize flag><program><param addr><param length><parameters><cr/lf>

where:

<wandID> - must be the same ID as sent in the parameter list to the **InitializeWand** function.

<reloadFlag> - 1 or 0 (ignored by DLL)

<initialize flag> - 1 or 0 (ignored by DLL)

<program> - the symbology file name to be loaded (is checked to be the same as the **symPath** parameter which is passed to **InitializeWand**). Like **symPath**, the **program** parameter must include the full path, for example: **c:\videx\3of9hcy.twp**. Long names in the path are acceptable, but spaces in the names will cause communication to fail.

<param addr> - must be 2816

<param length> - the number of parameters that will be sent to the wand. If loading a Hierarchy, this number is 19. If loading a Sequence, this number is 44.

<parameters> - The remaining parameters control the sequence in which the barcodes will be scanned.

For a hierarchy application:

*The first parameter represents the number of levels used in the hierarchy. For example, if the hierarchy consists of A, B, and C, then the first value would be 3. The 2nd through 17th parameters represent the level characters in the hierarchy. These parameters are the decimal values of the ASCII characters (i.e., the decimal value of the **A** character is **65**). If there is no character, use the value **255** as a placeholder. The alpha characters must be uppercase, as Code 3 of 9 does not support lowercase. The 18th parameter controls the ability to read bar codes outside of the hierarchy after the last level of the hierarchy is read. The value **00** enables this ability and **255** disables it. The 19th parameter controls whether the scanpad is turned on or off. The value **255** enables the scanpad and **00** disables it.*

For a sequence application:

*The first parameter, which is the same as in the hierarchy example, is the number of levels used in the sequence. The 2nd through 43rd parameters represent the level characters in the sequence. The 44th parameter controls the ability to have multiple reads of the last level of the hierarchy. The value **00** disables this ability and **255** enables it.*

For example, to program the wand with a hierarchy to read three levels of **A**, **B**, and **C**, with a wand ID of **1234**, a symbology of **3of9hcy.twp**, and with outside and scanpad reading enabled, the **aplPath** application file will contain:

1234 0 0 c:\videx\3of9hcy.twp 2816 19 3 65 66 67 255 255 255 255 255 255 255 255 255 255 255 255 255 00 255<cr><lf>

Declaration statements for Visual Basic:

```
Declare Function InitializeWand Lib "DLOAD32.DLL" (ByVal WandID As Long, ByVal symPath As String, ByVal aplPath As String, ByVal ComPortNumber As Long, ByVal initTime As Long, ByVal traceDebug As Long, ByVal IOdebug As Long) As Long

Declare Function DownLoadWand Lib "DLOAD32.DLL" (ByVal WandID As Long, ByVal DataFileName As String, ByVal ComPortNumber As Long, ByVal OverWriteData As Long, ByVal MemoryDump As Long, ByVal ClearMemory As Long, ByVal DLtime As Long, ByVal traceDebug As Long, ByVal IOdebug As Long) As Long

Declare Function DownLoadWandList Lib "DLOAD32.DLL" (ByVal WandIDList As String, ByVal DataFileName As String, ByVal ComPortNumber As Long, ByVal OverWriteData As Long, ByVal ClearMemory As Long, ByVal DLtime As Long, ByVal traceDebug As Long, ByVal IOdebug As Long) As Long

Declare Function SerOpen Lib "DLOAD32.DLL" (ByVal ComPortNumber As Long) As Long

Declare Function SerClose Lib "DLOAD32.DLL" (ByVal ComPortNumber As Long) As Long

Declare Function SetStatusMode Lib "DLOAD32.DLL" (ByVal statusMode As Long) As Long

Declare Sub SetProgress Lib "DLOAD32.DLL" (ByVal userFunction As Long)
```

Sample Function Calls for Visual Basic:

To initialize a wand with an ID of **1000** and a hierarchy application, the syntax would be:

returnValue = InitializeWand (1000, "c:\videx\3of9hcy.twp", "c:\videx\app.txt", 2, 70, 0, 0)

where the file **app.txt** contains:

1000 0 0 c:\videx\3of9hcy.twp 2816 19 3 67 84 69 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255

In this example, the wand is on com port 2, a timeout of 70 seconds is used, and the wand is programmed with a three-level hierarchy application with character values of **C**, **T**, and **E**.

To initialize the same wand to read Code 39 barcodes, but with no application, the syntax would be:

returnValue = InitializeWand (1000, "c:\videx\3of9kp.twp", "", 2, 70, 0, 0)

To download data from this wand, the syntax would be:

returnValue = DownLoadWand (1000, "data.bcd", 2, 0, 0, 1, 70, 0, 0)

The data from a wand with an ID of 1000 is appended to a file **data.bcd**. The wand is on com port 2, a timeout of 70 seconds is used, and the wand's data is cleared after a successful download.

To download data from three wands with ID's of 1000, 2000, and 4000, the syntax would be:

returnValue = DownLoadWandList ("wands.txt", "data.bcd", 2, 0, 1, 70, 0, 0)

where the file **wands.txt** contains:

1000 <cr> <lf> 2000 <cr> <lf> 4000 <cr> <lf>

The data from these three wands is appended to a file named **data.bcd**. The wands are on com port 2, a timeout of 70 seconds is used, and the data is cleared from the wands after a successful download.

To allow the user to cancel communication, it is necessary to define a callback function. This function will check for an event (for example, clicking on a command button), and then set a flag that will stop a communication in progress. The code to do this in Visual BASIC looks like:

```
Dim quitFlag as Boolean

Private Sub cmdDownloadWand_Click( )
    Dim returnValue as Long
    InitDownload
    returnValue = DownLoadWand ( 1000, "data.bcd", 2, 0, 0, 1, 70, 0, 0 )
    If returnValue < > 0 Then
        MsgBox "Download Error: " + Str (returnValue)
    Else
        MsgBox "Download Successful"
    End If
End Sub

Private Sub cmdCancelDownload_Click( )
    quitFlag = True
End Sub

Private Sub InitDownload( )
    quitFlag = False
    SetProgress AddressOf CancelCheck
End Sub

Public Function CancelCheck ( ByVal dummy1 As Long, ByVal dummy2 As _
Long ) As Long
    On Error Resume Next
    DoEvents
    If Form1.quitFlag Then
        CancelCheck = 1
    Else
        CancelCheck = 0
    End If
End Sub
```