Store Analog Functions

Class/Panel Name

Function Name

Store Analog Wfm I8

Store Analog Wfm F64 <u>niHWS_StoreAnalogWfmF64</u> niHWS_StoreAnalogWfml8 Store Analog Wfm I16 <u>niHWS_StoreAnalogWfmI16</u> Store Analog Wfm I32 <u>niHWS_StoreAnalogWfmI32</u>

niHWS_StoreAnalogWfmF64

C Function Prototype

tHWS_Status niHWS_StoreAnalogWfmF64 (const char filePath[], double rate, double gain, double offset, long compressionLevel, long numSamples, const double* wfmData, long overwriteExistingFile);

Creates a new file for storing double-precision floating-point analog waveform data.

This function is intended for storing a single waveform in a file. To store multiple waveforms in a file, use the <u>Low Level</u> functions.

Туре	Description
const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
double	The rate of the Sample clock used to capture the waveform.
double	Intended for waveforms to be generated by hardware that supports applying a gain control to the downloaded waveform. For example, this parameter can be used for the gain parameter in NI-FGEN.
	NI-HWS does not apply this gain when writing the data in wfmData .
double	Intended for waveforms to be generated by hardware that supports applying an offset control to the downloaded waveform. For example, this parameter can be used for the offset parameter in NI-FGEN.
	NI-HWS does not apply this offset when writing the data in wfmData .
long	Provides control over the amount of compression applied to the waveform samples as they are written to the HWS file.
	A value of 0 means no compression. A value of 9 means maximum compression. Values between zero and nine indicate increasing levels of compression.
long	The number of samples the new waveform contains.
const double*	The array containing the data for the waveform. This array should contain at least the amount of data indicated by the value assigned to numSamples .
long	Allows this function to overwrite the file if it exists.
	Set to 1 to overwrite the existing file and create a new file.
	Set to 0 to prevent overwriting an existing file.
	<pre>Type const char[] double double double long long long long long long long</pre>

<u>Return Value</u>

niHWS_StoreAnalogWfmI8

C Function Prototype

tHWS_Status niHWS_StoreAnalogWfmI8 (const char filePath[], double rate, double gain, double offset, long compressionLevel, long numSamples, const char* wfmData, long overwriteExistingFile);

Creates a new file for storing 8-bit signed integer (I8) analog waveform data.

This function is intended for storing a single waveform in a file. To store multiple waveforms in a file, use the <u>Low Level</u> functions.

Туре	Description
const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
double	The rate of the Sample clock used to capture the waveform.
double	Intended for waveforms to be generated by hardware that supports applying a gain control to the downloaded waveform. For example, this parameter can be used for the gain parameter in NI-FGEN.
	NI-HWS does not apply this gain when writing the data in wfmData .
double	Intended for waveforms to be generated by hardware that supports applying an offset control to the downloaded waveform. For example, this parameter can be used for the offset parameter in NI-FGEN.
	NI-HWS does not apply this offset when writing the data in wfmData .
long	Provides control over the amount of compression applied to the waveform samples as they are written to the HWS file.
	A value of 0 means no compression. A value of 9 means maximum compression. Values between zero and nine indicate increasing levels of compression.
long	The number of samples the new waveform contains.
const char*	The array containing the data for the waveform. This array should contain at least the amount of data indicated by the value assigned to numSamples .
long	Allows this function to overwrite the file if it exists.
	Set to 1 to overwrite the existing file and create a new file.
	Set to 0 to prevent overwriting an existing file.
	Type const char[] double double double long long const char*

<u>Return Value</u>

niHWS_StoreAnalogWfmI16

C Function Prototype

tHWS_Status niHWS_StoreAnalogWfmI16 (const char filePath[], double rate, double gain, double offset, long compressionLevel, long numSamples, const short* wfmData, long overwriteExistingFile);

Creates a new file for storing 16-bit signed integer (I16) analog waveform data.

This function is intended for storing a single waveform in a file. To store multiple waveforms in a file, use the <u>Low Level</u> functions.

Туре	Description
const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
double	The rate of the Sample clock used to capture the waveform.
double	Intended for waveforms to be generated by hardware that supports applying a gain control to the downloaded waveform. For example, this parameter can be used for the gain parameter in NI-FGEN.
	NI-HWS does not apply this gain when writing the data in wfmData .
double	Intended for waveforms to be generated by hardware that supports applying an offset control to the downloaded waveform. For example, this parameter can be used for the offset parameter in NI-FGEN.
	NI-HWS does not apply this offset when writing the data in wfmData .
long	Provides control over the amount of compression applied to the waveform samples as they are written to the HWS file.
	A value of 0 means no compression. A value of 9 means maximum compression. Values between zero and nine indicate increasing levels of compression.
long	The number of samples the new waveform contains.
const short*	The array containing the data for the waveform. This array should contain at least the amount of data indicated by the value assigned to numSamples .
long	Allows this function to overwrite the file if it exists.
	Set to 1 to overwrite the existing file and create a new file.
	Set to 0 to prevent overwriting an existing file.
	Type const char[] double double double long long const short*

<u>Return Value</u>

niHWS_StoreAnalogWfmI32

C Function Prototype

tHWS_Status niHWS_StoreAnalogWfmI32 (const char filePath[], double rate, double gain, double offset, long compressionLevel, long numSamples, const long* wfmData, long overwriteExistingFile);

Creates a new file for storing 32-bit signed integer (I32) analog waveform data.

This function is intended for storing a single waveform in a file. To store multiple waveforms in a file, use the <u>Low Level</u> functions.

Туре	Description
const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
double	The rate of the Sample clock used to capture the waveform.
double	Intended for waveforms to be generated by hardware that supports applying a gain control to the downloaded waveform. For example, this parameter can be used for the gain parameter in NI-FGEN.
	NI-HWS does not apply this gain when writing the data in wfmData .
double	Intended for waveforms to be generated by hardware that supports applying an offset control to the downloaded waveform. For example, this parameter can be used for the offset parameter in NI-FGEN.
	NI-HWS does not apply this offset when writing the data in wfmData .
long	Provides control over the amount of compression applied to the waveform samples as they are written to the HWS file.
	A value of 0 means no compression. A value of 9 means maximum compression. Values between zero and nine indicate increasing levels of compression.
long	The number of samples the new waveform contains.
const long*	The array containing the data for the waveform. This array should contain at least the amount of data indicated by the value assigned to numSamples .
long	Allows this function to overwrite the file if it exists.
	Set to 1 to overwrite the existing file and create a new file.
	Set to 0 to prevent overwriting an existing file.
	Type const char[] double double double long long const long*

<u>Return Value</u>

Retrieve Analog Functions

Class/Panel Name

Function Name

Retrieve Analog Wfm I8

Retrieve Analog Wfm F64 <u>niHWS_RetrieveAnalogWfmF64</u> niHWS_RetrieveAnalogWfml8 Retrieve Analog Wfm I16 niHWS_RetrieveAnalogWfmI16 Retrieve Analog Wfm I32 niHWS_RetrieveAnalogWfml32

niHWS_RetrieveAnalogWfmF64

C Function Prototype

tHWS_Status niHWS_RetrieveAnalogWfmF64 (const char filePath[], double* rate, double* gain, double* offset, long wfmBufferSize, double* wfmBuffer, long* numSamplesInWfm);

Retrieves the analog waveform as double-precision floating-point data from the file specified.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
rate	double*	The rate of the Sample clock used to capture the waveform.
gain	double*	This parameter is equivalent to element 1 of the coeffArray .
		NI-HWS stores polynomial scaling coefficients that are intended to be applied to waveform data by your application or your hardware. The offset and gain parameters in the high-level Store and Retrieve functions are identical to the first two elements of the coeffArray in the Get and Set Scaling Coefficients functions.
		Note If a waveform stored as integers is retrieved by this function, NI-HWS scales the integers to floating point numbers using the coeffArray and sets the gain parameter to 1.
offset	double*	This parameter is equivalent to element 0 of the coeffArray .
		NI-HWS stores polynomial scaling coefficients that are intended to be applied to waveform data by your application or your hardware. The offset and gain parameters in the high-level Store and Retrieve functions are identical to the first two elements of the coeffArray in the Get and Set Scaling Coefficients functions.
		Note If a waveform stored as integers is retrieved by this function, NI-HWS scales the integers to floating point numbers using the coeffArray and sets the offset parameter to 0.
wfmBufferSize	long	Specifies how many samples to read into wfmBuffer.
		If wfmBufferSize is 0, the total number of samples in the waveform is returned in numSamplesInWfm .
wfmBuffer	double*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to wfmBufferSize .
numSamplesInWfm	long*	Returns the total number of samples in the waveform.
<u>Return Value</u>		

niHWS_RetrieveAnalogWfml8

C Function Prototype

tHWS_Status niHWS_RetrieveAnalogWfmI8 (const char filePath[], double* rate, double* gain, double* offset, long wfmBufferSize, char* wfmBuffer, long* numSamplesInWfm);

Retrieves the analog waveform samples as 8-bit signed integers from the file specified.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
rate	double*	The rate of the Sample clock used to capture the waveform.
gain	double*	This parameter is equivalent to element 1 of the coeffArray .
		NI-HWS stores polynomial scaling coefficients that are intended to be applied to waveform data by your application or your hardware. The offset and gain parameters in the high-level Store and Retrieve functions are identical to the first two elements of the coeffArray in the Get and Set Scaling Coefficients functions.
offset	double*	This parameter is equivalent to element 0 of the coeffArray .
		NI-HWS stores polynomial scaling coefficients that are intended to be applied to waveform data by your application or your hardware. The offset and gain parameters in the high-level Store and Retrieve functions are identical to the first two elements of the coeffArray in the Get and Set Scaling Coefficients functions.
wfmBufferSize	long	Specifies how many samples to read into wfmBuffer.
		If wfmBufferSize is 0, the total number of samples in the waveform is returned in numSamplesInWfm .
wfmBuffer	char*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to wfmBufferSize .
numSamplesInWfm	long*	Returns the total number of samples in the waveform.
Return Value		

niHWS_RetrieveAnalogWfmI16

C Function Prototype

tHWS_Status niHWS_RetrieveAnalogWfmI16 (const char filePath[], double* rate, double* gain, double* offset, long wfmBufferSize, short* wfmBuffer, long* numSamplesInWfm);

Retrieves the analog waveform samples as 16-bit signed integers (I16) from the file specified.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
rate	double*	The rate of the Sample clock used to capture the waveform.
gain	double*	This parameter is equivalent to element 1 of the coeffArray .
		NI-HWS stores polynomial scaling coefficients that are intended to be applied to waveform data by your application or your hardware. The offset and gain parameters in the high-level Store and Retrieve functions are identical to the first two elements of the coeffArray in the Get and Set Scaling Coefficients functions.
offset	double*	This parameter is equivalent to element 0 of the coeffArray .
		NI-HWS stores polynomial scaling coefficients that are intended to be applied to waveform data by your application or your hardware. The offset and gain parameters in the high-level Store and Retrieve functions are identical to the first two elements of the coeffArray in the Get and Set Scaling Coefficients functions.
wfmBufferSize	long	Specifies how many samples to read into wfmBuffer.
		If wfmBufferSize is 0, the total number of samples in the waveform is returned in numSamplesInWfm .
wfmBuffer	short*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to wfmBufferSize .
numSamplesInWfm	long*	Returns the total number of samples in the waveform.
Return Value		

niHWS_RetrieveAnalogWfmI32

C Function Prototype

tHWS_Status niHWS_RetrieveAnalogWfmI32 (const char filePath[], double* rate, double* gain, double* offset, long wfmBufferSize, long* wfmBuffer, long* numSamplesInWfm);

Retrieves the analog waveform samples as 32-bit signed integers from the file specified.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
rate	double*	The rate of the Sample clock used to capture the waveform.
gain	double*	This parameter is equivalent to element 1 of the coeffArray .
		NI-HWS stores polynomial scaling coefficients that are intended to be applied to waveform data by your application or your hardware. The offset and gain parameters in the high-level Store and Retrieve functions are identical to the first two elements of the coeffArray in the Get and Set Scaling Coefficients functions.
offset	double*	This parameter is equivalent to element 0 of the coeffArray .
		NI-HWS stores polynomial scaling coefficients that are intended to be applied to waveform data by your application or your hardware. The offset and gain parameters in the high-level Store and Retrieve functions are identical to the first two elements of the coeffArray in the Get and Set Scaling Coefficients functions.
wfmBufferSize	long	Specifies how many samples to read into wfmBuffer.
		If wfmBufferSize is 0, the total number of samples in the waveform is returned in numSamplesInWfm .
wfmBuffer	long*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to wfmBufferSize .
numSamplesInWfm	long*	Returns the total number of samples in the waveform.
Return Value		

Store Digital Functions

Class/Panel Name

Function Name

Store Digital Wfm U8 Store Digital Wfm 2D U8 Store Digital Wfm U16 Store Digital Wfm U32 Store Digital WDT

niHWS_StoreDigitalWfmU8 niHWS_StoreDigitalWfm2DU8 niHWS_StoreDigitalWfmU16 Store Digital Wfm 2D U16 <u>niHWS_StoreDigitalWfm2DU16</u> niHWS_StoreDigitalWfmU32 Store Digital Wfm 2D U32 niHWS StoreDigitalWfm2DU32 niHWS StoreDigitalWfmWDT

niHWS_StoreDigitalWfmU8

C Function Prototype

tHWS_Status niHWS_StoreDigitalWfmU8 (const char filePath[], double rate, const char validSignals[], long compressionLevel, long numSamples, const unsigned char* wfmData, long overwriteExistingFile);

Creates a new file for storing digital 8-bit unsigned integer (U8) waveform data.

This function is intended for storing a single waveform in a file. To store multiple waveforms in a file, use the <u>Low Level</u> functions.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
rate	double	The rate of the Sample clock used to capture the waveform.
validSignals	const char[]	The list of lines (or bits) in the digital samples that are valid. For example, "0,3,6" means that only bits 0, 3, and 6 of each unsigned long digital sample contain valid digital data.
		The values in the string can receive any integer value between 0 and 7. Individual lines are separated by commas. You can use a hyphen to denote an inclusive range of lines. For example, "2-5" is equivalent to "2,3,4,5".
compressionLevel	long	This parameter provides control over the amount of compression applied to the waveform samples as they are written to the HWS file.
		A value of 0 means no compression. A value of 9 means maximum compression. Values between zero and nine indicate increasing levels of compression.
numSamples	long	The number of samples the new waveform contains.
wfmData	const unsigned char*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to numSamples .
overwriteExistingFile	long	Allows this function to overwrite the file if it exists.
		Set to 1 to overwrite the existing file and create a new file.
		Set to 0 to prevent overwriting an existing file.

<u>Return Value</u>

niHWS_StoreDigitalWfmU16

C Function Prototype

tHWS_Status niHWS_StoreDigitalWfmU16 (const char filePath[], double rate, const char validSignals[], long compressionLevel, long numSamples, const unsigned short* wfmData, long overwriteExistingFile);

Creates a new file for storing digital 16-bit unsigned (U16) waveform data.

This function is intended for storing a single waveform in a file. To store multiple waveforms in a file, use the <u>Low Level</u> functions.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
rate	double	The rate of the Sample clock used to capture the waveform.
validSignals	const char[]	The list of lines (or bits) in the digital samples that are valid. For example, "0,3,6" means that only bits 0, 3, and 6 of each unsigned long digital sample contain valid digital data.
		The values in the string can receive any integer value between 0 and 15. Individual lines are separated by commas. You can use a hyphen to denote an inclusive range of lines. For example, "2-5" is equivalent to "2,3,4,5".
compressionLevel	long	This parameter provides control over the amount of compression applied to the waveform samples as they are written to the HWS file.
		A value of 0 means no compression. A value of 9 means maximum compression. Values between zero and nine indicate increasing levels of compression.
numSamples	long	The number of samples the new waveform contains.
wfmData	const unsigned short*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to numSamples .
overwriteExistingFile	long	Allows this function to overwrite the file if it exists.
		Set to 1 to overwrite the existing file and create a new file.
		Set to 0 to prevent overwriting an existing file.

<u>Return Value</u>

niHWS_StoreDigitalWfmU32

C Function Prototype

tHWS_Status niHWS_StoreDigitalWfmU32 (const char filePath[], double rate, const char validSignals[], long compressionLevel, long numSamples, const unsigned long* wfmData, long overwriteExistingFile);

Creates a new file for storing digital 32-bit unsigned long (U32) waveform data.

This function is intended for storing a single waveform in a file. To store multiple waveforms in a file, use the <u>Low Level</u> functions.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
rate	double	The rate of the Sample clock used to capture the waveform.
validSignals	const char[]	The list of lines (or bits) in the digital samples that are valid. For example, "0,3,6" means that only bits 0, 3, and 6 of each unsigned long digital sample contain valid digital data.
		The values in the string can receive any integer value between 0 and 31. Individual lines are separated by commas. You can use a hyphen to denote an inclusive range of lines. For example, "2-5" is equivalent to "2,3,4,5".
compressionLevel	long	This parameter provides control over the amount of compression applied to the waveform samples as they are written to the HWS file.
		A value of 0 means no compression. A value of 9 means maximum compression. Values between zero and nine indicate increasing levels of compression.
numSamples	long	The number of samples the new waveform contains.
wfmData	const unsigned long*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to numSamples .
overwriteExistingFile	long	Allows this function to overwrite the file if it exists.
		Set to 1 to overwrite the existing file and create a new file.
		Set to 0 to prevent overwriting an existing file.

<u>Return Value</u>
niHWS_StoreDigitalWfm2DU8

Specific Function

C Function Prototype

tHWS_Status niHWS_StoreDigitalWfm2DU8 (const char filePath[], double rate, long compressionLevel, long numRows, long numSamplesPerRow, const unsigned char* wfmData, long overwriteExistingFile);

Creates a new file for storing a two-dimensional array of digital 8-bit unsigned integer data.

This function is intended for storing a single waveform in a file. To store multiple waveforms in a file use the <u>Low Level</u> functions.

Name	Туре	Description	
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.	
rate	double	The rate of the Sample clock used to capture the waveform.	
compressionLevel	long	This parameter provides control over the amount of compression applied to the waveform samples as they are written to the HWS file.	
		A value of 0 means no compression. A value of 9 means maximum compression. Values between zero and nine indicate increasing levels of compression.	
numRows	long	The number of rows the new waveform contains.	
numSamplesPerRow	long	The number of samples the new waveform contains.	
wfmData	const unsigned char*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to the numSamplesPerRow parameter times the number of signals.	
overwriteExistingFile	long	Allows this function to overwrite the file if it exists.	
		Set to 1 to overwrite the existing file and create a new file.	
		Set to 0 to prevent overwriting an existing file.	

niHWS_StoreDigitalWfm2DU16

Specific Function

C Function Prototype

tHWS_Status niHWS_StoreDigitalWfm2DU16 (const char filePath[], double rate, long compressionLevel, long numRows, long numSamplesPerRow, const unsigned short* wfmData, long overwriteExistingFile);

Creates a new file for storing a two-dimensional array of digital 16-bit (U16) data.

This function is intended for storing a single waveform in a file. To store multiple waveforms in a file use the <u>Low Level</u> functions.

Name	Туре	Description	
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.	
rate	double	The rate of the Sample clock used to capture the waveform.	
compressionLevel	long	This parameter provides control over the amount of compression applied to the waveform samples as they are written to the HWS file.	
		A value of 0 means no compression. A value of 9 means maximum compression. Values between zero and nine indicate increasing levels of compression.	
numRows	long	The number of rows the new waveform contains.	
numSamplesPerRow	long	The number of samples the new waveform contains.	
wfmData	const unsigned short*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to the numSamplesPerRow parameter times the number of rows.	
overwriteExistingFile	long	Allows this function to overwrite the file if it exists.	
		Set to 1 to overwrite the existing file and create a new file.	
		Set to 0 to prevent overwriting an existing file.	

niHWS_StoreDigitalWfm2DU32

Specific Function

C Function Prototype

tHWS_Status niHWS_StoreDigitalWfm2DU32 (const char filePath[], double rate, long compressionLevel, long numRows, long numSamplesPerRow, const unsigned long* wfmData, long overwriteExistingFile);

Creates a new file for storing a two-dimensional array of digital 32-bit (U32) data.

This function is intended for storing a single waveform in a file. To store multiple waveforms in a file use the <u>Low Level</u> functions.

Name	Туре	Description	
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.	
rate	double	The rate of the Sample clock used to capture the waveform.	
compressionLevel	long	This parameter provides control over the amount of compression applied to the waveform samples as they are written to the HWS file.	
		A value of 0 means no compression. A value of 9 means maximum compression. Values between zero and nine indicate increasing levels of compression.	
numRows	long	The number of rows the new waveform contains.	
numSamplesPerRow	long	The number of samples the new waveform contains.	
wfmData	const unsigned long*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to the numSamplesPerRow parameter times the number of rows.	
overwriteExistingFile	long	Allows this function to overwrite the file if it exists.	
		Set to 1 to overwrite the existing file and create a new file.	
		Set to 0 to prevent overwriting an existing file.	

niHWS_StoreDigitalWfmWDT

C Function Prototype

tHWS_Status niHWS_StoreDigitalWfmWDT (const char filePath[], double rate, long compressionLevel, tHWS_DigitalWDTDataLayout data_layout, long number_ofSignals, long number_ofSamples, const unsigned char* waveformData, long overwriteExistingFile);

Creates a new file for storing digital waveform data type (WDT) data.

This function is intended for storing a single waveform in a file. To store multiple waveforms in a file, use the <u>Low Level</u> functions.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
rate	double	The rate of the Sample clock used to capture the waveform.
compressionLevel	long	This parameter provides control over the amount of compression applied to the waveform samples as they are written to the HWS file.
		A value of 0 means no compression. A value of 9 means maximum compression. Values between zero and nine indicate increasing levels of compression.
dataLayout	tHWS_DigitalWDTDataLayout	Describes the layout of the waveform data.
		Defined Values
		niHWS_Val_GroupBySample—specifies that consecutive samples in wfmData are such that the array contains the first sample from every signal in the operation, then the second sample from every signal, up to the last sample from every signal. niHWS_Val_GroupBySignal—specifies that consecutive samples in wfmData are such that the array contains all the samples from the first signal in the operation, then all the samples from the second signal, up to all samples from the last signal.
numSignals	long	The number of signals the new waveform contains.
numSamples	long	The number of samples the new waveform contains.
wfmData	const unsigned char*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to the numSamples parameter times numSignals .
overwriteExistingFile	long	Allows this function to overwrite the file if it exists.
		Set to 1 to overwrite the existing file and create a new file.
		Set to 0 to prevent overwriting an existing file.

Retrieve Digital Functions

Class/Panel Name

Function Name

Retrieve Digital Wfm U8 Retrieve Digital Wfm 2D U8 Retrieve Digital Wfm U16 Retrieve Digital Wfm U32 Retrieve Digital Wfm WDT

niHWS RetrieveDigitalWfmU8 niHWS_RetrieveDigitalWfm2DU8 niHWS RetrieveDigitalWfmU16 Retrieve Digital Wfm 2D U16 <u>niHWS_RetrieveDigitalWfm2DU16</u> niHWS RetrieveDigitalWfmU32 Retrieve Digital Wfm 2D U32 niHWS RetrieveDigitalWfm2DU32 niHWS RetrieveDigitalWfmWDT

niHWS_RetrieveDigitalWfmU8

C Function Prototype

tHWS_Status niHWS_RetrieveDigitalWfmU8 (const char filePath[], double* rate, long validSignalsSize, char validSignals[], long wfmBufferSize, unsigned char* wfmBuffer, long* numSamplesInWfm);

Retrieves the digital waveform samples as 8-bit unsigned integers (U8) from the file specified.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
rate	double*	The rate of the Sample clock used to capture the waveform.
validSignalsSize	long	The number of characters that the string allocated for the validSignals parameter can hold.
		If validSignalsSize is 0 and no error occurs, the actual size of the validSignals string, including the end of string termination character, is returned in the status parameter.
validSignals	char[]	The list of lines (or bits) in the digital samples that are valid. For example, "0,3,6" means that only bits 0, 3, and 6 of each unsigned long digital sample contain valid digital data.
		The values in the string can receive any integer value between 0 and 7. Individual lines are separated by commas. You can use a hyphen to denote an inclusive range of lines. For example, "2-5" is equivalent to "2,3,4,5".
wfmBufferSize	long	Specifies how many samples to read into wfmBuffer.
		If wfmBufferSize is 0, the number of samples in the waveform is returned in numSamplesInWfm .
wfmBuffer	unsigned char*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to wfmBufferSize .
numSamplesInWfm	long*	Reports the total number of samples in the waveform.
<u>Return Value</u>		

niHWS_RetrieveDigitalWfmU16

C Function Prototype

tHWS_Status niHWS_RetrieveDigitalWfmU16 (const char filePath[], double* rate, long validSignalsSize, char validSignals[], long wfmBufferSize, unsigned short* wfmBuffer, long* numSamplesInWfm);

Retrieves the digital waveform samples as 16-bit unsigned integers (U16) from the file specified.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
rate	double*	The rate of the Sample clock used to capture the waveform.
validSignalsSize	long	The number of characters that the string allocated for the validSignals parameter can hold.
		If validSignalsSize is zero and no error occurs, the actual size of the validSignals string, including the end of string termination character, is returned in the status parameter.
validSignals	char[]	The list of lines (or bits) in the digital samples that are valid. For example, "0,3,6" means that only bits 0, 3, and 6 of each unsigned long digital sample contain valid digital data.
		The values in the string can receive any integer value between 0 and 15. Individual lines are separated by commas. You can use a hyphen to denote an inclusive range of lines. For example, "2-5" is equivalent to "2,3,4,5".
wfmBufferSize	long	Specifies how many samples to read into wfmBuffer.
		If wfmBufferSize is 0, the number of samples in the waveform is returned in numSamplesInWfm .
wfmBuffer	unsigned short*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to wfmBufferSize .
numSamplesInWfm	long*	Reports the total number of samples in the waveform.
<u>Return Value</u>		

niHWS_RetrieveDigitalWfmU32

C Function Prototype

tHWS_Status niHWS_RetrieveDigitalWfmU32 (const char filePath[], double* rate, long validSignalsSize, char validSignals[], long wfmBufferSize, unsigned long* wfmBuffer, long* numSamplesInWfm);

Retrieves the digital waveform samples as 32-bit unsigned integers (U32) from the file specified.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
rate	double*	The rate of the Sample clock used to capture the waveform.
validSignalsSize	long	The number of characters that the string allocated for the validSignals parameter can hold.
		If validSignalsSize is 0 and no error occurs, the actual size of the validSignals string, including the end of string termination character, is returned in the status parameter.
validSignals	char[]	The list of lines (or bits) in the digital samples that are valid. For example, "0,3,6" means that only bits 0, 3, and 6 of each unsigned long digital sample contain valid digital data.
		The values in the string can receive any integer value between 0 and 31. Individual lines are separated by commas. You can use a hyphen to denote an inclusive range of lines. For example, "2-5" is equivalent to "2,3,4,5".
wfmBufferSize	long	Specifies how many samples to read into wfmBuffer.
		If wfmBufferSize is 0, the number of samples in the waveform is returned in numSamplesInWfm .
wfmBuffer	unsigned long*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to wfmBufferSize .
numSamplesInWfm	long*	Reports the total number of samples in the waveform.
<u>Return Value</u>		

niHWS_RetrieveDigitalWfm2DU8

Specific Function

C Function Prototype

tHWS_Status niHWS_RetrieveDigitalWfm2DU8 (const char filePath[], double* rate, long wfmBufferSize, unsigned char* wfmBuffer, long* numRowsInWfm, long* numSamplesPerRowInWfm);

Retrieves the digital waveform samples as a two-dimensional array of 8bit unsigned integers (U8) from the file specified.

If there are multiple waveforms in the file, use the <u>low-level functions</u> to retrieve them.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for NI-HWS files, although using this extension is optional.
rate	double*	The rate of the Sample clock used to capture the waveform.
wfmBufferSize	long	Specifies how many samples to read into the wfmBuffer .
		If the wfmBufferSize is 0, the number of samples in the waveform is returned in the numSamplesPerRowInWfm parameter and the number of rows in the waveform is returned in the numRowsInWfm parameter.
wfmBuffer	unsigned char*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to the wfmBufferSize times the number of rows in the waveform.
numRowsInWfm	long*	Reports the number of rows the waveform.
numSamplesPerRowInWfm	long*	Reports the number of samples per row in the waveform.
		If the wfmBufferSize parameter is 0, the number of samples per row in the waveform is returned in the numSamplesPerRowInWfm parameter.
		If the numSamplesPerRowInWfm parameter is 0, the number of rows in the waveform is returned in the numRowsInWfm parameter.

niHWS_RetrieveDigitalWfm2DU16

Specific Function

C Function Prototype

tHWS_Status niHWS_RetrieveDigitalWfm2DU16 (const char filePath[], double* rate, long wfmBufferSize, unsigned short* wfmBuffer, long* numRowsInWfm, long* numSamplesPerRowInWfm);

Retrieves the digital waveform samples as a two-dimensional array of 16bit unsigned integers (U16) from the file specified.

If there are multiple waveforms in the file, use the <u>low-level functions</u> to retrieve them.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for NI-HWS files, although using this extension is optional.
rate	double*	The rate of the Sample clock used to capture the waveform.
wfmBufferSize	long	Specifies how many samples to read into the wfmBuffer .
		If the wfmBufferSize is 0, the number of samples in the waveform is returned in the numSamplesPerRowInWfm parameter and the number of rows in the waveform is returned in the numRowsInWfm parameter.
wfmBuffer	unsigned short*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to the wfmBufferSize times the number of rows in the waveform.
numRowsInWfm	long*	Reports the number of rows the waveform.
numSamplesPerRowInWfm	long*	Reports the number of samples per row in the waveform. If the WfmBufferSize parameter is 0, the number of samples per row in the waveform is returned in the numSamplesPerRowInWfm parameter.

niHWS_RetrieveDigitalWfm2DU32

Specific Function

C Function Prototype

tHWS_Status niHWS_RetrieveDigitalWfm2DU32 (const char filePath[], double* rate, long wfmBufferSize, unsigned long* wfmBuffer, long* numRowsInWfm, long* numSamplesPerRowInWfm);

Retrieves the digital waveform samples as a two-dimensional array of 32bit unsigned integers (U32) from the file specified.

If there are multiple waveforms in the file, use the <u>low-level functions</u> to retrieve them.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for NI-HWS files, although using this extension is optional.
rate	double*	The rate of the Sample clock used to capture the waveform.
wfmBufferSize	long	Specifies how many samples to read into the wfmBuffer .
		If the wfmBufferSize is 0, the number of samples in the waveform is returned in the numSamplesPerRowInWfm parameter and the number of rows in the waveform is returned in the numRowsInWfm parameter.
wfmBuffer	unsigned long*	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to the wfmBufferSize times the number of rows in the waveform.
numRowsInWfm	long*	Reports the number of rows the waveform.
numSamplesPerRowInWfm	long*	Reports the number of samples per row in the waveform.
		If the wfmBufferSize is 0, the number of samples in the waveform is returned in the numSamplesPerRowInWfm parameter and the number of rows in the waveform is returned in the numRowsInWfm parameter.

niHWS_RetrieveDigitalWfmWDT

Specific Function

C Function Prototype

tHWS_Status niHWS_RetrieveDigitalWfmWDT (const char filePath[], double *rate, tHWS_DigitalWDTDataLayout data_layout, long waveformBufferSize, unsigned char waveformBuffer[], long *number_ofSignals_inWaveform, long *number_ofSamples_inWaveform);

Retrieves the digital waveform data type samples from the file specified.

Name	Туре	Description
filePath	const char[]	Specifies the path and file name of the HWS file to create or open. The .hws extension is typically used for HWS files although using this extension is optional.
rate	double*	The rate of the sample clock used to capture the waveform.
dataLayout	tHWS_DigitalWDTDataLayout	Describes the layout of the waveform data.
		Defined Values
		niHWS_Val_GroupBySample— specifies that consecutive samples in waveformBuffer are such that the array contains the first sample from every signal in the operation, then the second sample from every signal, up to the last sample from every signal. niHWS_Val_GroupBySignal—specifies that consecutive samples in waveformBuffer are such that the array contains all the samples from the first signal in the operation, then all the samples from the second signal, up to all samples from the last signal.
waveformBufferSize	long	Specifies the size of waveformBuffer , in samples.
		Allocate enough memory for the entire waveform to be read, then pass the pre-allocated array and the size of that array to this function.
waveformBuffer	unsigned char[]	Contains the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to the Waveform Buffer Size parameter.
numberofSignalsinWaveform	long*	Reports the number of signals in the waveform.
		If the waveformBufferSize parameter is 0, the number of signals in the waveform is returned in the numberofSignalsinWaveform

parameter.

numberofSamplesinWaveform long*

Reports the number of samples in the waveform.

If the **waveformBufferSize** parameter is 0, the number of samples in the waveform is returned in the **numberofSamplesinWaveform** parameter.
Low-Level Functions

Class/Panel Name	Function Name
Open File	niHWS_OpenFile
Close File	niHWS_CloseFile
Read Functions	
Get Waveform Reference	niHWS_GetWfmReference
Read Analog Functions	
Read Analog F64	niHWS_ReadAnalogF64
Read Analog I8	niHWS_ReadAnalogI8
Read Analog I16	niHWS_ReadAnalogI16
Read Analog I32	niHWS_ReadAnalogI32
Read Digital Functions	
Read Digital WDT	niHWS_ReadDigitalWDT
Read Digital U8	niHWS_ReadDigitalU8
Read Digital 2D U8	niHWS_ReadDigital2DU8
Read Digital U16	niHWS_ReadDigitalU16
Read Digital 2D U16	niHWS_ReadDigital2DU16
Read Digital U32	niHWS_ReadDigitalU32
Read Digital 2D U32	niHWS_ReadDigital2DU32
Read Utility Functions	
Get Number of Groups	niHWS_GetNumberOfGroups
Get Group Name	niHWS_GetGroupName
Get Number of Waveforms	niHWS_GetNumberOfWaveforms
Get Waveform Name	niHWS_GetWfmName
Get Scaling Coefficients	niHWS_GetScalingCoefficients
Get Group String Attribute	niHWS_GetGroupStringAttribute
Get Waveform String Attribute	niHWS_GetWfmStringAttribute
Get Waveform I32 Attribute	niHWS_GetWfmI32Attribute
Get Waveform I64 Attribute	niHWS_GetWfmI64Attribute
Get Waveform F64 Attribute	niHWS_GetWfmF64Attribute
Write Functions	
New Waveform Reference	niHWS_NewWfmReference
Write Analog	
Write Analog F64	niHWS_WriteAnalogF64
Write Analog 18	niHWS_WriteAnalog18
Write Analog I16	niHWS_WriteAnalogI16

niHWS_WriteAnalogI32
niHWS_WriteDigitalWDT
niHWS_WriteDigitalU8
niHWS_WriteDigital2DU8
niHWS_WriteDigitalU16
niHWS_WriteDigital2DU16
niHWS_WriteDigitalU32
niHWS_WriteDigital2DU32
niHWS_SetScalingCoefficients
niHWS_SetGroupStringAttribute
niHWS_SetWfmStringAttribute
niHWS_SetWfmF64Attribute

Seek Get Error String niHWS_Seek niHWS_GetErrorString

niHWS_OpenFile

C Function Prototype

tHWS_Status niHWS_OpenFile (const char filePath[], tHWS_FilePermissions permissions, tHWS_FileHandle* fileHandle);

Opens or creates an HWS waveform file. The file must be closed with the \underline{niHWS} _CloseFile function.

Name	Туре	Description
filePath	const char[]	Specifies the path and filename of the HWS file to create or open. The .hws extension is typically used for HWS files, although using this extension is optional.
permissions	tHWS_FilePermissions	Specifies file access options
		Defined Values
		niHWS_Val_ReadOnly—Opens an existing file for reading. An error is returned if the specified file does not exist. An error is returned if an attempt is made to write to the file.
		niHWS_Val_ReadWriteExisting—Opens an existing file for both reading and writing. An error is returned if the specified file does not exist.
		niHWS_Val_ReadWriteCreateNew—Creates a new file for both reading and writing. An error is returned if the specified file already exists.
		niHWS_Val_ReadWriteCreateNewAlways—Creates a new file for both reading and writing. If the specified file already exists, its contents are lost.
fileHandle	tHWS_FileHandle*	Refers to an open HWS file.
Poturn Val	110	

<u>Return Value</u>

niHWS_CloseFile

C Function Prototype

tHWS_Status niHWS_CloseFile (tHWS_FileHandle fileHandle);

Closes the NI-HWS file specified by **fileHandle**.

NameTypeDescriptionfileHandletHWS_FileHandleRefers to an open HWS file.

<u>Return Value</u>

Read Functions

Class/Panel Name Get Waveform Reference **Read Analog Functions** Read Analog F64 Read Analog 18 Read Analog I16 Read Analog I32 **Read Digital Functions** Read Digital U8 Read Digital 2D U8 Read Digital U16 Read Digital 2D U16 Read Digital U32 Read Digital 2D U32 Read Digital WDT **Read Utility Functions** Get Number of Groups Get Group Name Get Number of Waveforms Get Waveform Name **Get Scaling Coefficients** Get Group String Attribute Get Waveform I32 Attribute

Get Waveform I64 Attribute

Get Waveform F64 Attribute

Function Name niHWS GetWfmReference niHWS ReadAnalogF64 niHWS ReadAnalogI8 niHWS ReadAnalogI16 niHWS ReadAnalogI32 niHWS ReadDigitalU8 niHWS ReadDigital2DU8 niHWS ReadDigitalU16 niHWS ReadDigital2DU16

niHWS ReadDigitalU32 niHWS ReadDigital2DU32 niHWS ReadDigitalWDT

niHWS GetNumberOfGroups niHWS_GetGroupName niHWS_GetNumberOfWaveforms niHWS_GetWfmName niHWS_GetScalingCoefficients niHWS GetGroupStringAttribute Get Waveform String Attribute niHWS GetWfmStringAttribute niHWS GetWfmI32Attribute niHWS GetWfmI64Attribute niHWS GetWfmF64Attribute

niHWS_GetWfmReference

C Function Prototype

tHWS_Status niHWS_GetWfmReference (tHWS_FileHandle fileHandle, const char groupName[], const char wfmName[], tHWS_WfmRef* wfmReference);

Returns a reference to an existing waveform.

Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.

Name	Туре	Description
fileHandle	tHWS_FileHandle	Refers to an open HWS file.
groupName	const char[]	The name of the group. Groups are identified by name for purposes of setting and getting group attributes.
		If only one group is in the file, this parameter can be NULL or the empty string.
wfmName	const char[]	The name of the waveform. If there is only one waveform in the group, the waveform name is optional (this parameter can be NULL or the empty string). Otherwise, you must supply the waveform name.
wfmReference	tHWS_WfmRef*	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
Return Value	<u>e</u>	

Read Analog Functions

Class/Panel Name	Function Name
Read Analog F64	niHWS_ReadAnalogF64
Read Analog I8	niHWS_ReadAnalogI8
Read Analog I16	niHWS_ReadAnalogI16
Read Analog I32	niHWS_ReadAnalogI32

niHWS_ReadAnalogF64

C Function Prototype

tHWS_Status niHWS_ReadAnalogF64 (tHWS_WfmRef wfmReference, long numSamplesToRead, double* wfmBuffer, long* actualSamplesRead);

Returns analog waveform samples as double-precision floating-point data beginning at the current read/write position.

Analog waveform data stored as doubles returns unaltered. Analog waveform data stored as integers (chars, shorts, or longs) is scaled to doubles using the stored scaling coefficients.

A read moves the read/write position so that it points to the next unread sample in the waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamplesToRead	long	Specifies how many samples to read into wfmBuffer .
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .
wfmBuffer	double*	The array that contains the data for the waveform. It is assumed that this array is big enough to contain at least the amount of data indicated by the value assigned to numSamplesToRead .
actualSamplesRead	long*	Reports the number of samples read.
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .

Return Value

niHWS_ReadAnalogI8

C Function Prototype

tHWS_Status niHWS_ReadAnalogI8 (tHWS_WfmRef wfmReference, long numSamplesToRead, char* wfmBuffer, long* actualSamplesRead);

Returns analog waveform samples as 8-bit signed integers (I8) beginning at the current read/write position. Only I8 data can be retrieved with this function.

A read moves the read/write position so that it points to the next unread sample in the waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamplesToRead	long	Specifies how many samples to read into wfmBuffer .
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .
wfmBuffer	char*	The array that contains the data for the waveform. It is assumed that this array is big enough to contain at least the amount of data indicated by the value assigned to numSamplesToRead .
actualSamplesRead	long*	Reports the number of samples read.
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .

Return Value

niHWS_ReadAnalogI16

C Function Prototype

tHWS_Status niHWS_ReadAnalogI16 (tHWS_WfmRef wfmReference, long numSamplesToRead, short* wfmBuffer, long* actualSamplesRead);

Returns analog waveform samples as 16-bit signed integers (I16) beginning at the current read/write position. Only data stored as 8- or 16-bit signed integers (chars or shorts) can be retrieved with this function. 8-bit data is sign-extended to 16 bits.

A read moves the read/write position so that it points to the next unread sample in the waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamplesToRead	long	Specifies how many samples to read into wfmBuffer .
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .
wfmBuffer	short*	The array that contains the data for the waveform. It is assumed that this array is big enough to contain at least the amount of data indicated by the value assigned to numSamplesToRead .
actualSamplesRead	long*	Reports the number of samples read.
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .

Return Value

niHWS_ReadAnalogI32

C Function Prototype

tHWS_Status niHWS_ReadAnalogI32 (tHWS_WfmRef wfmReference, long numSamplesToRead, long* wfmBuffer, long* actualSamplesRead);

Returns analog waveform samples as 32-bit signed integers (I32) beginning at the current read/write position. Data stored as 8- or 16-bit signed integers (chars or shorts) is sign-extended to 32 bits.

A read moves the read/write position so that it points to the next unread sample in the waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamplesToRead	long	Specifies how many samples to read into wfmBuffer .
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .
wfmBuffer	long*	The array that contains the data for the waveform. It is assumed that this array is big enough to contain at least the amount of data indicated by the value assigned to numSamplesToRead .
actualSamplesRead	long*	Reports the number of samples read.
		If numSamplesTo Read is 0, the number of samples in the waveform is returned in actualSamplesRead .

Return Value

Read Digital Functions

Class/Panel Name	Function Name
Read Digital U8	niHWS_ReadDigitalU8
Read Digital WDT	niHWS_ReadDigitalWDT
Read Digital 2D U8	niHWS_ReadDigital2DU8
Read Digital U16	niHWS_ReadDigitalU16
Read Digital 2D U16	niHWS_ReadDigital2DU16
Read Digital U32	niHWS_ReadDigitalU32
Read Digital 2D U32	niHWS_ReadDigital2DU32

niHWS_ReadDigitalU8

C Function Prototype

tHWS_Status niHWS_ReadDigitalU8 (tHWS_WfmRef wfmReference, long numSamplesToRead, unsigned char* wfmBuffer, long* actualSamplesRead);

Returns digital waveform samples as 8-bit unsigned integers (U8) beginning at the current read/write position.

A read moves the read/write position so that it points to the next unread sample in the waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamplesToRead	long	Specifies how many samples to read into wfmBuffer .
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .
wfmBuffer	unsigned char*	The array that contains the data for the waveform. It is assumed that this array is big enough to contain at least the amount of data indicated by the value assigned to numSamplesToRead .
actualSamplesRead	long*	Reports the number of samples read.
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .

Return Value

niHWS_ReadDigitalU16

C Function Prototype

tHWS_Status niHWS_ReadDigitalU16 (tHWS_WfmRef wfmReference, long numSamplesToRead, unsigned short* wfmBuffer, long* actualSamplesRead);

Returns digital waveform samples as 16-bit unsigned integers (U16) beginning at the current read/write position.

A read moves the read/write position so that it points to the next unread sample in the waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamplesToRead	long	Specifies how many samples to read into wfmBuffer .
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .
wfmBuffer	unsigned short*	The array that contains the data for the waveform. It is assumed that this array is big enough to contain at least the amount of data indicated by the value assigned to numSamplesToRead .
actualSamplesRead	long*	Reports the number of samples read.
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .

Return Value

niHWS_ReadDigitalU32

C Function Prototype

tHWS_Status niHWS_ReadDigitalU32 (tHWS_WfmRef wfmReference, long numSamplesToRead, unsigned long* wfmBuffer, long* actualSamplesRead);

Returns digital waveform samples as 32-bit unsigned integers (U32) beginning at the current read/write position.

A read moves the read/write position so that it points to the next unread sample in the waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamplesToRead	long	Specifies how many samples to read into wfmBuffer .
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .
wfmBuffer	unsigned long*	The array that contains the data for the waveform. It is assumed that this array is big enough to contain at least the amount of data indicated by the value assigned to numSamplesToRead .
actualSamplesRead	long*	Reports the number of samples read.
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .

Return Value

niHWS_ReadDigital2DU8

Specific Function

C Function Prototype

tHWS_Status niHWS_ReadDigital2DU8 (tHWS_WfmRef wfmReference, long numSamplesToReadPerRow, unsigned char* wfmBuffer, long* actualSamplesReadPerRow);
Returns digital waveform samples as a two-dimensional array of 8-bit unsigned integers (U8) beginning at the current read/write position.

A read moves the read/write position so that it points to the next unread sample in the waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamplesToReadPerRow	long	Specifies how many samples to read into the wfmBuffer .
		If the numSamplesToReadPerRow parameter is 0, the number of samples in the waveform is returned in the actualSamplesReadPerRow parameter.
wfmBuffer	unsigned char*	The array that contains the data for the waveform. It is assumed that this array is big enough to contain at least the amount of data indicated by the value assigned to the numSamplesToReadPerRow parameter times the number of rows of the 2D waveform. You can get the number of rows using the <u>niHWS_GetWfmI32Attribute</u> function to get the niHWS_Attr_NumRows attribute.
actualSamplesReadPerRow	long*	Reports the number of samples read per row. If the numSamplesToReadPerRow parameter is 0, the number of samples in the waveform is returned in the actualSamplesReadPerRow parameter.

niHWS_ReadDigital2DU16

Specific Function

C Function Prototype

tHWS_Status niHWS_ReadDigital2DU16 (tHWS_WfmRef wfmReference, long numSamplesToReadPerRow, unsigned short* wfmBuffer, long* actualSamplesReadPerRow);

Returns digital waveform samples as a two-dimensional array of 16-bit unsigned integers (U16) beginning at the current read/write position.

A read moves the read/write position so that it points to the next unread sample in the waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamplesToReadPerRow	long	Specifies how many samples to read into the wfmBuffer .
		If the numSamplesToReadPerRow parameter is 0, the number of samples in the waveform is returned in the actualSamplesReadPerRow parameter.
wfmBuffer	unsigned short*	The array that contains the data for the waveform. It is assumed that this array is big enough to contain at least the amount of data indicated by the value assigned to the numSamplesToReadPerRow parameter times the number of rows of the 2D waveform. You can get the number of rows using the <u>niHWS_GetWfmI32Attribute</u> function to get the niHWS_Attr_NumRows attribute.
actualSamplesReadPerRow	long*	Reports the number of samples read per row.
		If the numSamplesToReadPerRow parameter is 0, the number of samples in the waveform is returned in the actualSamplesReadPerRow parameter.

niHWS_ReadDigital2DU32

Specific Function

C Function Prototype

tHWS_Status niHWS_ReadDigital2DU32 (tHWS_WfmRef wfmReference, long numSamplesToReadPerRow, unsigned long* wfmBuffer, long* actualSamplesReadPerRow);

Returns digital waveform samples as a two-dimensional array of 32-bit unsigned integers (U32) beginning at the current read/write position.

A read moves the read/write position so that it points to the next unread sample in the waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamplesToReadPerRow	long	Specifies how many samples to read into the wfmBuffer .
		If the numSamplesToReadPerRow parameter is 0, the number of samples in the waveform is returned in the actualSamplesReadPerRow parameter.
wfmBuffer	unsigned long*	The array that contains the data for the waveform. It is assumed that this array is big enough to contain at least the amount of data indicated by the value assigned to the numSamplesToReadPerRow parameter times the number of rows of the 2D waveform. You can get the number of rows using the <u>niHWS_GetWfmI32Attribute</u> function to get the niHWS_Attr_NumRows attribute.
actualSamplesReadPerRow	long*	Reports the number of samples read per row. If the numSamplesToReadPerRow parameter is 0, the number of samples in the waveform is returned in the actualSamplesReadPerRow parameter.

niHWS_ReadDigitalWDT

C Function Prototype

tHWS_Status niHWS_ReadDigitalWDT (tHWS_WfmRef waveformReference, tHWS_DigitalWDTDataLayout data_layout, long number_ofSamples_toRead, unsigned char* waveformData, long *actualSamplesRead);

Returns digital waveform samples as a waveform data type, beginning at the current read/write position.

A read moves the read/write position so that it points to the next unread sample in the waveform.

The digital waveform is an array of chars where each char represents a single bit of a waveform. If **dataLayout** is configured for niHWS_Val_GroupBySample, then all the bits for the first sample are in the first elements of the array, all the bits for the next sample are in the next elements, and so on. If **dataLayout** is configured for niHWS_Val_GroupBySignal, then the first range of elements in the array contains the bit 0 values for each sample, the next set of array elements contain the bit 1 values for each sample, and so on.

Each bit is represented by a char because a bit can be any of the following eight different values: H, L, O, 1, X, Z, V, T.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
dataLayout	tHWS_DigitalWDTDataLayout	Describes the layout of the waveform data.
		Defined Values
		NIHWS_VAL_GROUP_BY_SAMPLE—specifies that consecutive samples in wfmBuffer are such that the array contains the first sample from every signal in the operation, then the second sample from every signal, up to the last sample from every signal. NIHWS_VAL_GROUP_BY_SIGNAL—specifies that consecutive samples in wfmBuffer are such that the array contains all the samples from the first signal in the operation, then all the samples from the second signal, up to all samples from the last signal.
numSamplesToRead	long	Specifies how many samples to read into wfmBuffer .
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .
wfmBuffer	unsigned char*	The array that contains the data for the waveform. It is assumed that this array is big enough to contain at least the amount of data indicated by the value assigned to numSamplesToRead times the total number of signals.
actualSamplesRead	long*	Reports the number of samples read.
		If numSamplesToRead is 0, the number of samples in the waveform is returned in actualSamplesRead .

Read Utility Functions

Class/Panel Name

Function Name

Get Number of Groups Get Group Name Get Number of Waveforms Get Waveform Name Get Scaling Coefficients Get Group String Attribute Get Waveform I32 Attribute Get Waveform I64 Attribute Get Waveform F64 Attribute

niHWS GetNumberOfGroups niHWS_GetGroupName niHWS GetNumberOfWaveforms <u>niHWS_GetWfmName</u> niHWS GetScalingCoefficients niHWS GetGroupStringAttribute Get Waveform String Attribute niHWS GetWfmStringAttribute niHWS GetWfmI32Attribute niHWS_GetWfmI64Attribute niHWS GetWfmF64Attribute

niHWS_GetNumberOfGroups

C Function Prototype

tHWS_Status niHWS_GetNumberOfGroups (tHWS_FileHandle fileHandle, long* numberOfGroups);

Reports the number of waveform groups contained in the NI-HWS file.

NameTypeDescriptionfileHandletHWS_FileHandleRefers to an open HWS file.numberOfGroupslong*The number of waveform groups in the NI-HWS file.

niHWS_GetGroupName

C Function Prototype

tHWS_Status niHWS_GetGroupName (tHWS_FileHandle fileHandle, long groupIndex, long groupNameSize, char groupName[]);

Returns the name of a group in an NI-HWS file. Groups are identified by name for purposes of setting and getting group attributes.

Name	Туре	Description
fileHandle	tHWS_FileHandle	Refers to an open HWS file.
groupIndex	long	A number such that $0 \le $ groupIndex < numberOfGroups
		where
		numberOfGroups is obtained by calling <u>niHWS_GetNumberOfGroups</u> . If five groups are in a file, their group indices are 0 through 4. The only use for the groupIndex parameter is to obtain the name of a group. Groups are identified by name for purposes of setting and getting group attributes.
groupNameSize	long	Specifies the number of characters in the groupName.
		If this parameter is set to 0 and there is no error, the actual number of characters, including the end of string termination character, in the groupName are returned in the status parameter.
groupName	char[]	The name of the group specified by the groupIndex parameter. Groups are identified by name for purposes of setting and getting group attributes.
		If groupName is not NULL, up to groupNameSize characters are copied into it.
		If the actual size of groupName is larger, an error is returned along with the portion of the name that fits.

niHWS_GetNumberOfWaveforms

C Function Prototype

tHWS_Status niHWS_GetNumberOfWaveforms (tHWS_FileHandle fileHandle, const char groupName[], long* numberOfWaveforms);

Reports the number of waveforms in the group specified by **groupName**.

If there is only one group in the HWS file, the **groupName** parameter can be NULL or the empty string.

Name	Туре	Description
fileHandle	tHWS_FileHandle	Refers to an open HWS file.
groupName	const char[]	The name of the group. Groups are identified by name for purposes of setting and getting group attributes.
		If only one group is in the file groupName can be NULL or the empty string.
numberOfWaveforms	long*	Returns the number of waveforms in the group specified by groupName .

niHWS_GetWfmName

C Function Prototype

tHWS_Status niHWS_GetWfmName (tHWS_FileHandle fileHandle, const char groupName[], long wfmIndex, long wfmNameSize, char wfmName[]);

Returns the name of a waveform in a group. The name of a waveform is necessary for getting a reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.

Name	Туре	Description
fileHandle	tHWS_FileHandle	Refers to an open HWS file.
groupName	const char[]	The name of the group. Groups are identified by name for purposes of setting and getting group attributes.
		If only one group is in the file, groupName can be NULL or the empty string.
wfmIndex	long	A number such that 0 ≤ wfmIndex < numberOfWaveforms
		where
		numberOfWaveforms is obtained by calling <u>niHWS_GetNumberOfWaveforms</u> . If five waveforms are in a group, their waveform indices are 0 through 4.
		The only use for the wfmIndex parameter is to obtain the name of a waveform.
wfmNameSize	long	Specifies the number of characters in the wfmName parameter.
		If this parameter is set to 0 and no error occurs, the actual number of characters, including the end of string termination character, in the wfmName are returned in the status parameter.
wfmName	char[]	The name of the waveform specified by the wfmIndex parameter.
		The name of a waveform is necessary for getting a reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
		If wfmName is not NULL, up to wfmNameSize characters are copied into it.
		If the actual size of the waveform name is larger, an error is returned along with the portion of the name that fits.

niHWS_GetScalingCoefficients

C Function Prototype

tHWS_Status niHWS_GetScalingCoefficients (tHWS_WfmRef wfmReference, long numCoeffs, double* coeffArray, long* actualNumCoeffs);

Retrieves the coefficients of an *n*th degree polynomial scaling function.

When an analog waveform stored as integer data is retrieved by a floating-point read or retrieve function, the data is scaled according to this polynomial scaling function.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numCoeffs	long	The number of coefficients to return in coeffArray.
		If numCoeffs is 0, the number of coefficients in the polynomial are returned in the actualNumCoeffs .
		If the number of coefficients in the polynomial is greater than the nonzero value of numCoeffs , then as many coefficients as can fit are returned in the coeffArray along with an error.
		The actual number of coefficients returned in the coeffArray is always returned in actualNumCoeffs .
coeffArray	double*	The array of polynomial scaling coefficients. The coefficient for the <i>n</i> th term of the polynomial is returned in the <i>n</i> th index of the coeffArray .
		For example, the coefficients for a 2nd-degree polynomial,
		$y = ax^2 + bx + c$
		would be
		coeffArray[0] = c coeffArray[1] = b coeffArray[2] = a
actualNumCoeffs	long*	The actual number of coefficients returned in coeffArray .
		If numCoeffs is 0, the number of coefficients in the polynomial are returned in actualNumCoeffs .
		If the number of coefficients in the polynomial is greater than the nonzero value of numCoeffs , then as many coefficients as can fit are returned in the coeffArray parameter along with an error.

niHWS_GetGroupStringAttribute

C Function Prototype

tHWS_Status niHWS_GetGroupStringAttribute (tHWS_FileHandle fileHandle, const char groupName[], tHWS_GroupStringAttributes attributeID, long stringSize, char attributeVal[]);

Retrieves text attributes that apply to all the waveforms within a group.

Name	Туре	Description
fileHandle	tHWS_FileHandle	Refers to an open HWS file.
groupName	const char[]	The name of the group. Groups are identified by name for purposes of setting and getting group attributes.
		If only one group is in the file, groupName can be NULL or the empty string.
attributeID	tHWS_GroupStringAttributes	Specifies the attribute to retrieve.
		Defined Values
		niHWS_Attr_GroupNote—A general purpose text field for any comment that applies to all the waveforms stored in this group.
		niHWS_Attr_Technician—A general purpose text field intended for the name of the technician who acquired or generated the waveforms in this group.
		niHWS_Attr_Project—A general purpose text field intended for any information about the project for which the waveforms in this group were created.
		niHWS_Attr_GroupTimestamp—A string of the form hour:minutes:seconds.fractional seconds followed by AM or PM MM/DD/YYYY. This string is intended to refer to all the waveforms in the group.
stringSize	long	The number of characters that the string allocated for the attributeVal parameter can hold.
		If stringSize is 0 and no error occurs, the actual size of the string attribute, including the end of string termination character, is returned in the status parameter.
attributeVal	char[]	The current setting of the attribute specified by attributeID.
Return Va	llue	

niHWS_GetWfmStringAttribute

C Function Prototype

tHWS_Status niHWS_GetWfmStringAttribute (tHWS_WfmRef wfmReference, tHWS_WfmStringAttributes attributeID, long stringSize, char attributeVal[]);

Retrieves text attributes that apply to the waveform specified by **wfmReference**.

Name	Туре	Description
wfmReference	tHWS_WfmRef	References to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
attributeID	tHWS_WfmStringAttributes	Selects the attribute to retrieve.
		Defined Values
		niHWS_Attr_WfmNote— A general purpose text field for any information pertaining to the waveform.
		niHWS_Attr_XAxisLabel—A general purpose text field intended for labeling the x-axis of a waveform data plot.
		niHWS_Attr_YAxisLabel—A general purpose text field intended for labeling the y-axis of a waveform data plot.
		niHWS_Attr_XAxisUnits—A general purpose text field intended for denoting the units of the X Axis Increment attribute.
		niHWS_Attr_YAxisUnits—A general purpose text field intended for denoting the units of the waveform data.
		niHWS_Attr_WfmName—A read-only attribute that is the same as that returned by <u>niHWS_GetWfmName</u> .
		niHWS_AttrValid_Signals—The list of lines (or bits) in the samples of a digital waveform that are valid. For example, "0,3,6" means that only bits 0, 3, and 6 of each unsigned long digital sample contain valid digital data. A hyphen may be used to denote an inclusive range of consecutive lines. For example, "2-5" is equivalent to "2, 3, 4, 5".
		niHWS_Attr_WfmTimestamp—A string of the form hour:minutes:seconds.fractional seconds followed by AM or PM MM/DD/YYYY. Intended to serve as the time the waveform was originally acquired.
stringSize	long	The number of characters that the string allocated for attributeVal can hold.
		If stringSize is 0 and no error occurs, the actual size of the string attribute, including the end of string termination character, is returned in the status parameter.
attributeVal	char[]	The current setting of the attribute specified by attributeID .
<u>Return Valu</u>	<u>e</u>	

niHWS_GetWfmI32Attribute

C Function Prototype

tHWS_Status niHWS_GetWfmI32Attribute (tHWS_WfmRef wfmReference, tHWS_WfmI32Attributes attributeID, long* attributeVal);

Retrieves integer attributes that apply to the waveform specified by **wfmReference**.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
attributeID	tHWS_WfmI32Attributes	The waveform attribute to retrieve.
		niHWS_Attr_YAxisDataType— A read-only attribute that returns the type of integer or floating-point data that the waveform is stored as.
		Defined Values
		niHWS_Val_DataTypeI8 niHWS_Val_DataTypeI16 niHWS_Val_DataTypeI32 niHWS_Val_DataTypeF64 niHWS_Val_DataTypeU32 niHWS_Val_DataTypeDWDT niHWS_Val_DataTypeU8 niHWS_Val_DataTypeU16
		niHWS_Attr_IOType—A read-only attribute that returns the type of the waveform as either analog (niHWS_Val_Analog), digital (niHWS_Val_Digital), or unknown (niHWS_Val_Unknown).
		niHWS_Attr_ScalingType—A read-only attribute that returns the only currently supported scaling type of niHWS_Val_Polynomial.
		niHWS_Attr_WaveformSize—A read-only attribute that returns the total number of samples in a waveform. For digital 2D files with multiple rows, this attribute refers to the number of samples per row.
		$niHWS_Attr_NumRows_A$ read-only attribute that returns the number of rows in the waveform.
		niHWS_Attr_NumSignals—A read-only attribute that returns the number of signals in the waveform.
attributeVal	long*	The current setting of the attribute specified by attributeID .
niHWS_GetWfmI64Attribute

C Function Prototype

tHWS_Status niHWS_GetWfmI64Attribute (tHWS_WfmRef wfmReference, tHWS_WfmI64Attributes attributeID, __int64* attributeVal);

Retrieves 64-bit signed integer attributes that apply to the waveform specified by **wfmReference**.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
attributeID	tHWS_WfmI64Attributes	The waveform attribute to retrieve.
		Defined Value
		niHWS_Attr_WaveformSizeI64—The number of samples in the waveform.
attributeVal	int64*	The current setting of the attribute specified by attributeID .
Return Value	<u>e</u>	

niHWS_GetWfmF64Attribute

C Function Prototype

tHWS_Status niHWS_GetWfmF64Attribute (tHWS_WfmRef wfmReference, tHWS_WfmF64Attributes attributeID, double* attributeVal);

Retrieves floating point attributes that apply to the waveform specified by **wfmReference**.

Name	Туре	Description	
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.	
attributeID	tHWS_WfmF64Attributes	The waveform attribute to retrieve.	
		Defined Values	
		niHWS_Attr_XAxisStart—The X value for the first point in the waveform. The Scope Soft Front Panel stores relativeInitialX in this attribute, for example.	
		niHWS_Attr_XAxisIncrement—The period between samples in the waveform. The rate of the Sample clock used to capture the waveform is the reciprocal of the x-axis increment.	
		niHWS_Attr_WaveformSize—A read-only attribute that returns the total number of samples in a waveform.	
		niHWS_Attr_WfmYAxisMin—The minimum value of the y- axis after applying the scaling coefficients.	
		niHWS_Attr_WfmYAxisMax—The maximum value of the y- axis after applying the scaling coefficients.	
attributeVal	double*	The current setting of the attribute specified by attributeID parameter.	
Return Value			

Write Functions

Class/Panel Name	Function Name
New Waveform Reference	niHWS_NewWfmReference
Write Analog Functions	
Write Analog F64	niHWS_WriteAnalogF64
Write Analog I8	niHWS_WriteAnalog18
Write Analog I16	niHWS_WriteAnalogI16
Write Analog I32	niHWS_WriteAnalogI32
Write Digital Functions	
Write Digital WDT	niHWS_WriteDigitalWDT
Write Digital U8	niHWS_WriteDigitalU8
Write Digital 2D U8	niHWS_WriteDigital2DU8
Write Digital U16	niHWS_WriteDigitalU16
Write Digital 2D U16	niHWS_WriteDigital2DU16
Write Digital U32	niHWS_WriteDigitalU32
Write Digital 2D U32	niHWS_WriteDigital2DU32
Write Utility Functions	
Set Scaling Coefficients	niHWS_SetScalingCoefficien
Set Group String Attribute	niHWS_SetGroupStringAttrib
Set Waveform String Attribute	niHWS_SetWfmStringAttribu

<u>nts</u> oute <u>ite</u> Set Waveform F64 Attribute <u>niHWS_SetWfmF64Attribute</u>

niHWS_NewWfmReference

C Function Prototype

tHWS_Status niHWS_NewWfmReference (tHWS_FileHandle fileHandle, const char groupName[], const char wfmName[], long compressionLevel, tHWS_WfmRef* wfmReference);

Creates a new waveform in the specified group and returns a reference to it.

Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.

Name	Туре	Description
fileHandle	tHWS_FileHandle	Refers to an open HWS file.
groupName	const char[]	Identifies the group that the new waveform is created in.
		If the group referred to by the groupName parameter does not exist, then a new group is created and given the name assigned to groupName .
		groupName can be NULL or the empty string under the following conditions:
		 The file is new and without groups. A new group is created, and the new waveform is created in that group.
		Only one group is in the file. The new waveform is created in that group.
wfmName	const char[]	The name of the waveform. If this waveform is the only one in the group, then the name is optional (it can be NULL or the empty string). Otherwise, you must supply a name when creating a new waveform.
compressionLevel	long	Provides control over the amount of compression applied to the waveform samples as they are written to the HWS file.
		A value of 0 means no compression. A value of 9 means maximum compression. Values between zero and nine indicate increasing levels of compression.
wfmReference	tHWS_WfmRef*	References the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and for reading and writing data.
<u>Return Value</u>		

Write Analog Functions

Class/Panel Name	
Write Analog F64	r
Write Analog I8	r
Write Analog I16	r
Write Analog I32	r

Function Name

hiHWS_WriteAnalogF64

niHWS_WriteAnalog18

niHWS_WriteAnalogI16

niHWS_WriteAnalogI32

niHWS_WriteAnalogF64

C Function Prototype

tHWS_Status niHWS_WriteAnalogF64 (tHWS_WfmRef wfmReference, long numSamples, const double* wfmData);

Writes double-precision floating-point analog waveform data into the waveform beginning at the current read/write position.

A write moves the read/write position so that it points to the next sample to be written. Repeated writes to the same waveform concatenate data to that waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	References the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamples	long	The number of samples in wfmData to write to the file.
wfmData	const double*	The array containing the data for the waveform. It is assumed that this array contains at least the amount of data indicated by the value assigned to numSamples .

Return Value

niHWS_WriteAnalogI8

C Function Prototype

tHWS_Status niHWS_WriteAnalogI8 (tHWS_WfmRef wfmReference, long numSamples, const char* wfmData);

Writes 8-bit signed integer (I8) analog waveform data into the waveform beginning at the current read/write position.

A write moves the read/write position such that it points to the next sample. Repeated writes to the same waveform concatenate data to that waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamples	long	The number of samples in wfmData to write to the file.
wfmData	const char*	The array containing the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to numSamples .
	_	

<u>Return Value</u>

niHWS_WriteAnalogI16

C Function Prototype

tHWS_Status niHWS_WriteAnalogI16 (tHWS_WfmRef wfmReference, long numSamples, const short* wfmData);

Writes 16-bit signed integer (I16) analog waveform data into the waveform beginning at the current read/write position.

A write moves the read/write position such that it points to the next sample. Repeated writes to the same waveform concatenate data to that waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamples	long	The number of samples in wfmData to write to the file.
wfmData	const short*	The array containing the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to numSamples .
	_	

<u>Return Value</u>

niHWS_WriteAnalogI32

C Function Prototype

tHWS_Status niHWS_WriteAnalogI32 (tHWS_WfmRef wfmReference, long numSamples, const long* wfmData);

Writes 32-bit signed integer (I32) analog waveform data into the waveform beginning at the current read/write position.

A write moves the read/write position such that it points to the next sample. Repeated writes to the same waveform concatenate data to that waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamples	long	The number of samples in wfmData to write to the file.
wfmData	const long*	The array containing the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to numSamples .
	_	

<u>Return Value</u>

Write Digital Functions

Class/Panel Name	Function Name
Write Digital U8	niHWS_WriteDigitalU8
Write Digital 2D U8	niHWS_WriteDigital2DU8
Write Digital U16	niHWS_WriteDigitalU16
Write Digital 2D U16	niHWS_WriteDigital2DU16
Write Digital U32	niHWS_WriteDigitalU32
Write Digital 2D U32	niHWS_WriteDigital2DU32
Write Digital WDT	niHWS_WriteDigitalWDT

niHWS_WriteDigitalU8

C Function Prototype

tHWS_Status niHWS_WriteDigitalU8 (tHWS_WfmRef wfmReference, long numSamples, const unsigned char* wfmData);

Writes digital 8-bit unsigned integer (U8) waveform data into the waveform at the current read/write position.

A write moves the read/write position such that it points to the next sample. Repeated writes to the same waveform concatenate data to that waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamples	long	The number of samples in wfmData to write to the file.
wfmData	const unsigned char*	The array containing the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to numSamples .

Return Value

niHWS_WriteDigitalU16

C Function Prototype

tHWS_Status niHWS_WriteDigitalU16 (tHWS_WfmRef wfmReference, long numSamples, const unsigned short* wfmData);

Writes digital 16-bit unsigned integer (U16) waveform data into the waveform at the current read/write position.

A write moves the read/write position such that it points to the next sample. Repeated writes to the same waveform concatenate data to that waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamples	long	The number of samples in wfmData to write to the file.
wfmData	const unsigned short*	The array containing the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to numSamples .

Return Value

niHWS_WriteDigitalU32

C Function Prototype

tHWS_Status niHWS_WriteDigitalU32 (tHWS_WfmRef wfmReference, long numSamples, const unsigned long* wfmData);

Writes digital 32-bit unsigned integer (U32) waveform data into the waveform at the current read/write position.

A write moves the read/write position such that it points to the next sample. Repeated writes to the same waveform concatenate data to that waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numSamples	long	The number of samples in wfmData to write to the file.
wfmData	const unsigned long*	The array containing the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to numSamples .

Return Value

niHWS_WriteDigital2DU8

Specific Function

C Function Prototype

tHWS_Status niHWS_WriteDigital2DU8 (tHWS_WfmRef wfmReference, long numRows, long numSamplesPerRow, const unsigned char* wfmData);

Writes a two-dimensional array of digital 8-bit unsigned integer (U8) waveform data into the waveform beginning at the current read/write position.

A write moves the read/write position such that it points to the next sample. Repeated writes to the same waveform concatenate data to that waveform.

When you call this function, the number of rows in the array must be equal to the number of rows in niHWS_Attr_ValidSignals.

If you have an existing waveform that contains data, the number of rows in the new data must equal the number of rows in the existing data.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numRows	long	The number of rows in the wfmData to write to the file.
numSamplesPerRow	long	The number of samples per row to write to the file.
wfmData	const unsigned char*	The array containing the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to the numSamplesPerRow parameter times the number of rows of the 2D waveform. You can get the number of rows using the <u>niHWS_GetWfmI32Attribute</u> function to get the niHWS_Attr_NumRows attribute.

Return Value
niHWS_WriteDigital2DU16

Specific Function

C Function Prototype

tHWS_Status niHWS_WriteDigital2DU16 (tHWS_WfmRef wfmReference, long numRows, long numSamplesPerRow, const unsigned short* wfmData);

Writes a two-dimensional array of digital 16-bit unsigned integer (U16) waveform data into the waveform beginning at the current read/write position.

A write moves the read/write position such that it points to the next sample. Repeated writes to the same waveform concatenate data to that waveform.

When you call this function, the number of rows in the array must be equal to the number of rows in niHWS_Attr_ValidSignals.

If you have an existing waveform that contains data, the number of rows in the new data must equal the number of rows in the existing data.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numRows	long	The number of rows in the $\ensuremath{\textit{wfmData}}$ to write to the file.
numSamplesPerRow	long	The number of samples per row to write to the file.
wfmData	const unsigned short*	The array containing the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to the numSamplesPerRow parameter times the number of rows of the 2D waveform. You can get the number of rows using the <u>niHWS_GetWfmI32Attribute</u> function to get the niHWS_Attr_NumRows attribute.

Return Value

niHWS_WriteDigital2DU32

Specific Function

C Function Prototype

tHWS_Status niHWS_WriteDigital2DU32 (tHWS_WfmRef wfmReference, long numRows, long numSamplesPerRow, const unsigned long* wfmData);

Writes a two-dimensional array of digital 32-bit unsigned integer (U32) waveform data into the waveform beginning at the current read/write position.

A write moves the read/write position such that it points to the next sample. Repeated writes to the same waveform concatenate data to that waveform.

When you call this function, the number of rows in the array must be equal to the number of rows in niHWS_Attr_ValidSignals.

If you have an existing waveform that contains data, the number of rows in the new data must equal the number of rows in the existing data.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numRows	long	The number of rows in the wfmData to write to the file.
numSamplesPerRow	long	The number of samples per row to write to the file.
wfmData	const unsigned long*	The array containing the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to the numSamplesPerRow parameter times the number of rows of the 2D waveform. You can get the number of rows using the <u>niHWS_GetWfmI32Attribute</u> function to get the niHWS_Attr_NumRows attribute.

Return Value

niHWS_WriteDigitalWDT

C Function Prototype

tHWS_Status niHWS_WriteDigitalWDT (tHWS_WfmRef waveformReference, tHWS_DigitalWDTDataLayout data_layout, long number_ofSignals, long number_ofSamples, const unsigned char* waveformData);

Writes waveform data type data into the waveform at the current read/write position.

A write moves the read/write position such that it points to the next sample. Repeated writes to the same waveform concatenate data to that waveform.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
dataLayout	tHWS_DigitalWDTDataLayout	Describes the layout of the waveform data.
		Defined Values
		niHWS_Val_GroupBySample—specifies that consecutive samples in wfmData are such that the array contains the first sample from every signal in the operation, then the second sample from every signal, up to the last sample from every signal. niHWS_Val_GroupBySignal—specifies that consecutive samples in wfmData are such that the array contains all the samples from the first signal in the operation, then all the samples from the second signal, up to all samples from the last signal.
numSignals	long	The number of signals in wfmData to write to the file.
numSamples	long	The number of samples in wfmData to write to the file.
wfmData	const unsigned char*	The array containing the data for the waveform. This array must be large enough to contain at least the amount of data indicated by the value assigned to numSamples times the value of numSignals .

Return Value

Write Utility Functions

Class/Panel Name

Function Name

Set Scaling Coefficients Set Group String Attribute Set Waveform F64 Attribute

niHWS_SetScalingCoefficients niHWS_SetGroupStringAttribute Set Waveform String Attribute <u>niHWS_SetWfmStringAttribute</u> niHWS_SetWfmF64Attribute

niHWS_SetScalingCoefficients

C Function Prototype

tHWS_Status niHWS_SetScalingCoefficients (tHWS_WfmRef wfmReference, long numCoeffs, const double* coeffArray);

Establishes an *n*th degree polynomial scaling function.

When an analog waveform stored as integer data is retrieved by a floating-point Read or Retrieve function, NI-HWS scales the data according to this polynomial scaling function.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
numCoeffs	long	The number of coefficients in coeffArray .
coeffArray	const double*	The array of polynomial scaling coefficients. The coefficient for the n th term of the polynomial is contained in the n th index of coeffArray .
		For example, the coefficients for a 2nd degree polynomial,
		$y = ax^2 + bx + c$
		would be
		coeffArray[0] = c coeffArray[1] = b coeffArray[2] = a

Return Value

niHWS_SetGroupStringAttribute

C Function Prototype

tHWS_Status niHWS_SetGroupStringAttribute (tHWS_FileHandle fileHandle, const char groupName[], tHWS_GroupStringAttributes attributeID, const char attributeVal[]);

Sets the value of text attributes that apply to all the waveforms within a group.

Name	Туре	Description
fileHandle	tHWS_FileHandle	Refers to an open HWS file.
groupName	const char[]	The name of the group. Groups are identified by name for purposes of setting and getting group attributes.
		If only one group is in the file this parameter can be NULL or the empty string.
attributeID	tHWS_GroupStringAttributes	Selects the attribute to set.
		Defined Values
		niHWS_Attr_GroupNote—A general purpose text field for any comment that applies to all the waveforms stored in this group.
		niHWS_Attr_Technician—A general purpose text field intended for the name of the technician who acquired or generated the waveforms in this group.
		niHWS_Attr_Project—A general purpose text field intended for any information about the project for which the waveforms in this group were created.
		niHWS_Attr_GroupTimestamp—A string of the form hour:minutes:seconds.fractional seconds followed by AM or PM MM/DD/YYYY. This string is intended to refer to all the waveforms in the group.
attributeVal	const char[]	The new setting of the attribute specified by attributeID .

Return Value

niHWS_SetWfmStringAttribute

C Function Prototype

tHWS_Status niHWS_SetWfmStringAttribute (tHWS_WfmRef wfmReference, tHWS_WfmStringAttributes attributeID, const char attributeVal[]);

Sets the value of text attributes that apply to the waveform specified by **wfmReference**.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
attributeID	tHWS_WfmStringAttributes	Specifies the attribute to set.
		Defined Values
		niHWS_Attr_WfmNote— A general purpose text field for any information pertaining to the waveform.
		niHWS_Attr_XAxisLabel—A general purpose text field intended for labeling the x-axis of a waveform data plot.
		niHWS_Attr_YAxisLabel—A general purpose text field intended for labeling the y-axis of a waveform data plot.
		niHWS_Attr_XAxisUnits—A general purpose text field intended for denoting the units of the X Axis Increment attribute.
		niHWS_Attr_YAxisUnits—A general purpose text field intended for denoting the units of the waveform data.
		niHWS_Attr_ValidSignals—The list of lines (or bits) in the samples of a digital waveform that are valid. For example, "0,3,6" means that only bits 0, 3, and 6 of each unsigned long digital sample contain valid digital data. A hyphen may be used to denote an inclusive range of consecutive lines. For example, "2-5" is equivalent to "2, 3, 4, 5".
		niHWS_Attr_WfmTimestamp—A string of the form hour:minutes:seconds.fractional seconds followed by AM or PM MM/DD/YYYY. Intended to serve as the time the waveform was originally acquired.
attributeVal	const char[]	The new setting of the attribute specified by attributeID.

<u>Return Value</u>

niHWS_SetWfmF64Attribute

C Function Prototype

tHWS_Status niHWS_SetWfmF64Attribute (tHWS_WfmRef wfmReference, tHWS_WfmF64Attributes attributeID, double attributeVal);

Sets floating point attributes that apply to the waveform specified by **wfmReference**.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
attributeID	tHWS_WfmF64Attributes	The waveform attribute to set.
		Defined Values
		niHWS_Attr_XAxisStart—The X value for the first point in the waveform. The Scope Soft Front Panel stores relativeInitialX in this attribute for example.
		niHWS_Attr_XAxisIncrement—The period between samples in the waveform. The rate of the Sample clock used to capture the waveform is the reciprocal of the x-axis increment.
		niHWS_Attr_WfmYAxisMin—The minimum value of the y- axis after applying the scaling coefficients.
		niHWS_Attr_WfmYAxisMax—The maximum value of the y- axis after applying the scaling coefficients.
attributeVal	double	The new setting of the attribute specified by attributeID .
<u>Return Valu</u>	<u>e</u>	

niHWS_Seek

C Function Prototype

tHWS_Status niHWS_Seek (tHWS_WfmRef wfmReference, tHWS_RelativeTo relativeTo, long offset);

There is a single position within each waveform for both reading and writing. This function moves this position. First, the position is moved to the place in the waveform specified by the **relativeTo** parameter, then the **offset** is added.

Name	Туре	Description
wfmReference	tHWS_WfmRef	A reference to the waveform. Waveforms are identified by reference for purposes of setting and getting waveform attributes and reading and writing data.
relativeTo	tHWS_RelativeTo	Specifies the initial placement of the read/write position prior to adding offset .
		Defined Values
		niHWS_Start—Before the first sample in the waveform niHWS_End—After the last sample in the waveform niHWS_CurrentPosition—The current sample in the waveform
offset	long	After the read/write position is moved according to the value of the relativeTo parameter, the value of offset is added to determine the final read/write position.
		offset can be negative. Having offset as negative is useful when relativeTo is at the End of the waveform.
Return Valu	<u>e</u>	

niHWS_GetErrorString

Specific Function

C Function Prototype

tHWS_Status niHWS_GetErrorString (tHWS_Status errorCode, long stringSize, char errorString);

This function provides the error description for the specified error code.

Name	Туре	Description
errorCode	tHWS_Status	Error code for which the function provides a description.
stringSize	long	The number of characters that the string allocated for the errorString parameter can hold.
		If stringSize is 0 and no error occurs, the actual size of the error string, including the end of string termination character, is returned in the status parameter.
errorString	char[]	The description for the provided error code.

Return Value

Reports the status of this operation. To obtain a text description of the status code, call $\underline{niHWS}_GetErrorString$.

The general meaning of the status code is as follows:

Value	Meaning
0	Success
Negative Values	Errors