Numerix Host Library

Reference Manual

Version 3.40

4 August, 2018

Email : numerix@numerix-dsp.com
WWW : http://www.numerix-dsp.com

DOCUMENTATION CONVENTIONS 4

INTRODUCTION 5

Rebuilding the Library 5

ASCII TEXT OUTPUT FUNCTIONS 6
print_matrix 6
print_buffer 7
print_polar 8
print_rectangular 9

FILE I/O FUNCTIONS 10

Data File Formats 10

BIN File Functions 11
bin_read_data 11
bin_write_data 12

CSV File Functions 13
csv_read_data 13
csv_write_data 14

DAT File Functions 15
dat_read_data 15
dat_write_data 16
dat_read_header 17
dat_write_header 18

SIG File Functions 19
sig_read_data 19
sig_write_data 20

WAV File Functions 21
wav_read_data 22
wav_write_data 23
wav_read_word 24
wav_read_long 25
wav_write_word 26
wav_write_long 27
wav_read_header 28
wav_write_header 29
wav_display_info 30
wav_set_info 31
wav_file_length 32
wav_read_file 33
wav_write_file 34

wav_write_file_scaled 35

XMT File Functions **36**
xmt_read_data 36

Documentation Conventions

The documentation uses the following conventions :

The ANSI C standard conventions have been followed, for example hexadecimal numbers are prefixed by '0x'.

Names of directories, files and functions are given in italics.

Important programming information is indicated with the symbol : 

Introduction

The Numerix Host Library functions include simple text and file I/O functionality that are designed to aid in the development of DSP applications. These functions are designed to be used in conjunction with the SigLib DSP library but can be used without it. They have been developed under Windows and Linux but will re-compile on almost any processor that supports stdio.h functionality through it's compiler and JTAG debug facilities.

Updates to this library are available from available from http://www.numerix-dsp.com/files.

The standard functions accept arrays of data of type "double".

Rebuilding the Library

To rebuild the library you can use the following batch / make / project files :

GCC (Cygwin) under Windows	gcc_win_buildlib.bat
Microsoft Visual C/C++ (32 bit)	mbuildlib.bat
Microsoft Visual C/C++ (64 bit)	mbuildlib_64.bat
UNIX / Linux	makefile.lx
OSX	makefile.macos

ASCII Text Output Functions

The ASCII output functions are located in the file dspconio.c and will work on any processor that supports console I/O functionality via stdio.h.

print_matrix

FUNCTION NAME

print_matrix

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void print_matrix (const double *,	Pointer to matrix
const int,	Number of rows in matrix
const int)	Number of columns in matrix

FUNCTION DESCRIPTION

Print the contents of a 2D matrix onto the screen.

NOTES ON USE

This is very useful for debugging.

FUNCTION CROSS REFERENCE

print_buffer.

FUNCTION NAME

print_buffer

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void print_buffer (const double *,	Pointer to data buffer
const int)	Buffer size

FUNCTION DESCRIPTION

Print the contents of a buffer onto the screen one sample at a time.

NOTES ON USE

This is very useful for debugging.

FUNCTION CROSS REFERENCE

print_matrix.

FUNCTION NAME

print_polar

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void print_polar (const ComplexPolar_s *) Complex variable

FUNCTION DESCRIPTION

Print out the complex polar variable.

NOTES ON USE

This function is implemented as a macro.

The complex variable is defined as :

```
typedef struct
{
       SFLOAT magn;
       SFLOAT angle;
} ComplexPolar_s;
```

FUNCTION CROSS REFERENCE

print_rectangular.

FUNCTION NAME

print_rectangular

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void print_rectangular (const ComplexRect_s *) Complex variable

FUNCTION DESCRIPTION

Print out the complex rectangular variable.

NOTES ON USE

This function is implemented as a macro.

The complex variable is defined as :

```
typedef struct
{
       SFLOAT real;
       SFLOAT imag;
} ComplexRect_s;
```

FUNCTION CROSS REFERENCE

print_polar.

File I/O functions

The Numerix Host Library includes a range of functions for storing data, in floating point format, to hard disk. The functions treat the data in blocks and there are functions for reading and writing the data. The file read functions will zero pad any buffers if there is not sufficient data in the remainder of the file to fill the buffer.

Data File Formats

The library supports single channel file I/O in the following formats :

File Extension	Description
.bin	Contiguous 16 bit binary data
.csv	Comma Separated Variable format for importing into a spreadsheet
.dat	Two column format, with header. Column 1 : sample timestamp Column 2 : data sample This format is used by gnuplot
.sig	A single column of floating point numbers that represent the data sequence
.wav	16 bit .wav file

BIN File Functions

The following functions are used to read and write .bin files.

Data is stored in contiguous 16 bit data format and can be either little ‘l’ or big ‘b’ endian, as defined by the endian mode parameter.

bin_read_data

FUNCTION NAME

bin_read_data

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int bin_read_data (double *,	Destination data pointer
FILE *,	File pointer
const char,	Endian mode,
const int)	Buffer length

FUNCTION DESCRIPTION

This function reads a buffer of floating-point data from the disk.

NOTES ON USE

This function operates in a stream oriented mode and will read successive blocks of data from the file until the end of the file is reached.

This function will zero pad any buffers if there is not sufficient data in the remainder of the file to fill the buffer.

The file must be opened prior to using this function.

The function returns the number of samples read from the file.

FUNCTION CROSS REFERENCE

bin_write_data.

FUNCTION NAME

bin_write_data

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void bin_write_data (const double *,	Source data pointer
FILE *,	File pointer
const char,	Endian mode,
const int)	Buffer length

FUNCTION DESCRIPTION

This function writes a buffer of floating-point data to the disk.

NOTES ON USE

This function operates in a stream oriented mode and will append successive blocks of to the end of the file.

The file must be opened prior to using this function.

FUNCTION CROSS REFERENCE

bin_read_data.

CSV File Functions

The following functions are used to read and write .csv files.

csv_read_data

FUNCTION NAME

csv_read_data

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int csv_read_data (double *,	Destination data pointer
FILE *,	File pointer
const int)	Buffer length

FUNCTION DESCRIPTION

This function reads a buffer of floating-point data from the disk.

NOTES ON USE

This function operates in a stream oriented mode and will read successive blocks of data from the file until the end of the file is reached.

This function will zero pad any buffers if there is not sufficient data in the remainder of the file to fill the buffer.

The file must be opened prior to using this function.

The function returns the number of samples read from the file.

FUNCTION CROSS REFERENCE

csv_write_data.

FUNCTION NAME

csv_write_data

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void csv_write_data (const double *,	Source data pointer
FILE *,	File pointer
const int)	Buffer length

FUNCTION DESCRIPTION

This function writes a buffer of floating-point data to the disk.

NOTES ON USE

This function operates in a stream oriented mode and will append successive blocks of to the end of the file.

The file must be opened prior to using this function.

FUNCTION CROSS REFERENCE

csv_read_data.

DAT File Functions

The following functions are used to read and write .dat files.

These functions write files that are compatible with Gnuplot.

The file write functions require that the sample rate is passed as a parameter.

dat_read_data

FUNCTION NAME

dat_read_data

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int dat_read_data (double *,	Destination data pointer
FILE *,	File pointer
const int)	Buffer length

FUNCTION DESCRIPTION

This function reads a buffer of floating-point data from the disk.

NOTES ON USE

This function operates in a stream oriented mode and will read successive blocks of data from the file until the end of the file is reached.

This function will zero pad any buffers if there is not sufficient data in the remainder of the file to fill the buffer.

The file must be opened prior to using this function.

The function returns the number of samples read from the file.

FUNCTION CROSS REFERENCE

dat_write_data, dat_read_header, dat_write_header.

FUNCTION NAME

dat_write_data

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void dat_write_data (const double *,	Source data pointer
FILE *,	File pointer
const double sampleRate,	Sample rate
const int sampleIndex,	Sample index
const int)	Buffer length

FUNCTION DESCRIPTION

This function writes a buffer of floating-point data to the disk.

NOTES ON USE

This function operates in a stream oriented mode and will append successive blocks of to the end of the file.

The sample index parameter is used to maintain the index across successive writes.

The file must be opened prior to using this function.

FUNCTION CROSS REFERENCE

dat_read_data, dat_read_header, dat_write_header.

FUNCTION NAME

dat_read_header

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

double dat_read_header (FILE *) File pointer

FUNCTION DESCRIPTION

The dat_read_header function reads the header information from a dat file and returns the sample rate.

NOTES ON USE

The file must be opened prior to using this function.

FUNCTION CROSS REFERENCE

dat_read_data, dat_write_data, dat_write_header

FUNCTION NAME

dat_write_header

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void dat_write_header (FILE *,	File pointer
const double)	Sample rate

FUNCTION DESCRIPTION

The dat_write_header function writes the sample rate to the dat file header.

NOTES ON USE

The file must be opened prior to using this function.

FUNCTION CROSS REFERENCE

dat_read_data, dat_write_data, dat_read_header

SIG File Functions

The following functions are used to read and write .sig files.

The data is formatted in a single column.

sig_read_data

FUNCTION NAME

sig_read_data

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int sig_read_data (double *,	Destination data pointer
FILE *,	File pointer
const int)	Buffer length

FUNCTION DESCRIPTION

This function reads a buffer of floating-point data from the disk.

NOTES ON USE

This function operates in a stream oriented mode and will read successive blocks of data from the file until the end of the file is reached.

This function will zero pad any buffers if there is not sufficient data in the remainder of the file to fill the buffer.

The file must be opened prior to using this function.

The function returns the number of samples read from the file.

FUNCTION CROSS REFERENCE

sig_write_data.

FUNCTION NAME

sig_write_data

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void sig_write_data (const double *,	Source data pointer
FILE *,	File pointer
const int)	Buffer length

FUNCTION DESCRIPTION

This function writes a buffer of floating-point data to the disk.

NOTES ON USE

This function operates in a stream oriented mode and will append successive blocks of to the end of the file.

The file must be opened prior to using this function.

FUNCTION CROSS REFERENCE

sig_read_data.

WAV File Functions

The following functions are used to read and write .wav files. These functions require a structure of type WAV_FILE_INFO, which is defined as :

```
typedef struct
{
        int     SampleRate;
        int     NumberOfSamples;
        int     NumberOfChannels;
        int     WordLength;
        int     BytesPerSample;
        int     DataFormat;
} WAV_FILE_INFO;
```

This structure can be accessed directly from any program however functions are supplied for reading and writing to it.

Note : when writing a stream to a .wav file it is first necessary to write the header using the function wav_write_header () then the data can be written to the file. Once all of the data has been written and the exact number of samples is known then the number of samples can be re-written to the header and the function wav_write_header should be called again.

For multi-channel wav files, the data is returned with the channels multiplexed into a single array so the array length must equal the NumberOfSamples*NumberOfChannels. The SigLib DSP library includes functions for multiplexing and de-multiplexing data streams.

FUNCTION NAME

wav_read_data

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int wav_read_data (double *,	Destination data pointer
FILE *,	File pointer
const WAV_FILE_INFO,	Wave file information structure
const int)	Buffer length

FUNCTION DESCRIPTION

The wav_read_data function reads a buffer of wave file data from the disk.

NOTES ON USE

This function operates in a stream oriented mode and will read successive blocks of data from the file until the end of the file is reached.

This function will zero pad any buffers if there is not sufficient data in the remainder of the file to fill the buffer.

The function returns the number of samples read from the file.

The file must be opened prior to using this function.

Returns WavInfo.NumberOfSamples = 0 on error.

FUNCTION CROSS REFERENCE

wav_write_data, wav_read_word, wav_read_long, wav_write_word, wav_write_long, wav_read_header, wav_write_header, wav_display_info, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

FUNCTION NAME

wav_write_data

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void wav_write_data (const double *,	Source data pointer
FILE *,	File pointer
const WAV_FILE_INFO,	Wave file information structure
const int)	Buffer length

FUNCTION DESCRIPTION

The wav_write_data function writes a buffer of wave file data to the disk.

NOTES ON USE

This function operates in a stream oriented mode and will append successive blocks of to the end of the file.

The file must be opened prior to using this function.

FUNCTION CROSS REFERENCE

wav_read_data, wav_read_word, wav_read_long, wav_write_word, wav_write_long, wav_read_header, wav_write_header, wav_display_info, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

FUNCTION NAME

wav_read_word

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int wav_read_word (FILE *) File pointer

FUNCTION DESCRIPTION

The wav_read_word function reads a word of data from a wave file.

The file must be opened prior to using this function.

NOTES ON USE

The function returns the word read from the file.

FUNCTION CROSS REFERENCE

wav_read_data, wav_write_data, wav_read_long, wav_write_word, wav_write_long, wav_read_header, wav_write_header, wav_display_info, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

FUNCTION NAME

wav_read_long

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int wav_read_long (FILE *) File pointer

FUNCTION DESCRIPTION

The wav_read_long function reads an int word of data from a wave file.

NOTES ON USE

The function returns the int word read from the file.

The file must be opened prior to using this function.

FUNCTION CROSS REFERENCE

wav_read_data, wav_write_data, wav_read_word, wav_write_word, wav_write_long, wav_read_header, wav_write_header, wav_display_info, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

FUNCTION NAME

wav_write_word

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void wav_write_word (const int,	Data word to write
FILE *)	File pointer

FUNCTION DESCRIPTION

The wav_write_word function writes a word of data to the disk.

NOTES ON USE

The file must be opened prior to using this function.

FUNCTION CROSS REFERENCE

wav_read_data, wav_write_data, wav_read_word, wav_read_long, wav_write_long, wav_read_header, wav_write_header, wav_display_info, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

FUNCTION NAME

wav_write_long

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void wav_write_long (const int,	Long data word to write
FILE *)	File pointer

FUNCTION DESCRIPTION

The wav_write_long function writes a int word of data to the disk.

NOTES ON USE

The file must be opened prior to using this function.

FUNCTION CROSS REFERENCE

wav_read_data, wav_write_data, wav_read_word, wav_read_long, wav_write_word, wav_read_header, wav_write_header, wav_display_info, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

FUNCTION NAME

wav_read_header

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

WAV_FILE_INFO wav_read_header (FILE *) File pointer

FUNCTION DESCRIPTION

The wav_read_header function reads the header information from a wave file and returns it in the WAV_FILE_INFO structure.

NOTES ON USE

The file must be opened prior to using this function.

Returns WavInfo.NumberOfSamples = 0 on error.

FUNCTION CROSS REFERENCE

wav_read_data, wav_write_data, wav_read_word, wav_read_long, wav_write_word, wav_write_long, wav_write_header, wav_display_info, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

FUNCTION NAME

wav_write_header

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void wav_write_header (FILE *,	File pointer
const WAV_FILE_INFO)	Wave file information structure

FUNCTION DESCRIPTION

The wav_write_header function writes the header information to a wave file from the WAV_FILE_INFO structure.

NOTES ON USE

The file must be opened prior to using this function.

FUNCTION CROSS REFERENCE

wav_read_data, wav_write_data, wav_read_word, wav_read_long, wav_write_word, wav_write_long, wav_read_header, wav_display_info, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

FUNCTION NAME

wav_display_info

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

void wav_display_info (const WAV_FILE_INFO) Wave file information structure

FUNCTION DESCRIPTION

The wav_display_info function prints out the header information stored in the WAV_FILE_INFO structure.

NOTES ON USE

FUNCTION CROSS REFERENCE

wav_read_data, wav_write_data, wav_read_word, wav_read_long, wav_write_word, wav_write_long, wav_read_header, wav_write_header, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

FUNCTION NAME

wav_set_info

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

WAV_FILE_INFO wav_set_info (const int,	Sample rate
const int,	Number of samples
const int,	Number of channels
const int,	Word length
const int,	Bytes per sample
const int)	Data format

FUNCTION DESCRIPTION

The wav_set_info function generates a WAV_FILE_INFO structure from the supplied data.

NOTES ON USE

FUNCTION CROSS REFERENCE

wav_read_data, wav_write_data, wav_read_word, wav_read_long, wav_write_word, wav_write_long, wav_read_header, wav_write_header, wav_display_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

FUNCTION NAME

wav_file_length

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int wav_file_length (const char *) Filename

FUNCTION DESCRIPTION

This function returns the number of samples in the .wav file.

NOTES ON USE

FUNCTION CROSS REFERENCE

wav_write_data, wav_read_word, wav_read_long, wav_write_word, wav_write_long, wav_read_header, wav_write_header, wav_display_info, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled

FUNCTION NAME

wav_read_file

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

WAV_FILE_INFO wav_read_data (double *, Destination data pointer
const char *) Filename

FUNCTION DESCRIPTION

This function reads the contents of the .wav file data from the disk.

NOTES ON USE

It is important to ensure that the destination array is long enough to receive the data.

Returns the WAV_FILE_INFO structure for the data read, with the number of samples read set to -1 on file read error.

FUNCTION CROSS REFERENCE

wav_write_data, wav_read_word, wav_read_long, wav_write_word, wav_write_long, wav_read_header, wav_write_header, wav_display_info, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

FUNCTION NAME

wav_write_file

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int wav_write_file (double *,	Destination data pointer
const char *,	Filename
const WAV_FILE_INFO,	Wave file information structure
const int)	Buffer length

FUNCTION DESCRIPTION

This function writes the contents of the array to the .wav file.

NOTES ON USE

It is important to ensure that the destination array is long enough to receive the data.

Returns the number of samples written, -1 for file open error.

FUNCTION CROSS REFERENCE

wav_write_data, wav_read_word, wav_read_long, wav_write_word, wav_write_long, wav_read_header, wav_write_header, wav_display_info, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

FUNCTION NAME

wav_write_file_scaled

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int wav_write_file_scaled (double *,	Destination data pointer
const char *,	Filename
const WAV_FILE_INFO,	Wave file information structure
const int)	Buffer length

FUNCTION DESCRIPTION

This function writes the contents of the array to the .wav file. The output is scaled to a magnitude of 32767.0

NOTES ON USE

It is important to ensure that the destination array is long enough to receive the data.

Returns the number of samples written, -1 for file open error.

FUNCTION CROSS REFERENCE

wav_write_data, wav_read_word, wav_read_long, wav_write_word, wav_write_long, wav_read_header, wav_write_header, wav_display_info, wav_set_info, wav_file_length, wav_read_file, wav_write_file, wav_write_file_scaled.

XMT File Functions

The following function is used to read a XMOS xTIMEcomposer .xmt files.

xmt_read_data

FUNCTION NAME

xmt_read_data

FUNCTION PROTOTYPE AND PARAMETER DESCRIPTION

int xmt_read_data (double *,	Destination data pointer
FILE *,	File pointer
const long)	Buffer length

FUNCTION DESCRIPTION

This function reads a buffer of floating-point data from the disk.

NOTES ON USE

This function operates in a stream oriented mode and will read successive blocks of data from the file until the end of the file is reached.

This function will zero pad any buffers if there is not sufficient data in the remainder of the file to fill the buffer.

The file must be opened prior to using this function.

The function returns the number of samples read from the file.

FUNCTION CROSS REFERENCE