HYPERLABS™
Innovation at Hyperspeed since 1992

HYPERLABS ZTDR™

DLL Programmer's Guide (v 1.1.x)

Revision 1.1.0 | www.hyperlabsinc.com

Introduction

These brief reference materials apply to the ZTDR™ DLL v1.x.x, compatible with all HL11xx TDR instruments from HYPERLABS.

This document is a work in progress, and is provided on an as-is basis. It is ultimately the responsibility of the end-user to integrate the DLL in their application.

For further technical support, please visit our website: www.hyperlabsinc.com/support.aspx

Table of Contents

Section 1: Preface

The ZTDR™ DLL provides users with a streamlined way to harness the HL11xx TDR functionality outside of the normal Windows® software environment.

Only functions intended to be user-facing are documented here. Although the DLL contains more functions than are listed here, HYPERLABS strongly recommends against trying to use them ad-hoc.

Section 2: Dependencies

The ZTDR™ DLL and HL11xx hardware are dependent on the FTDI drivers distributed with ZTDR™. When working with the DLL, the following files should all be accessible to the complier:

- FTD2XX.dll
- FTD2XX.h
- FTD2XX.lib
- ZTDR_1XX.dll
- ZTDR_1XX.h
- ZTDR_1XX.lib

Section 3: ZTDR™ DLL Test Program

HYPERLABS has also created a ZTDR™ DLL Test program. This program, written in Visual C++, provides the programmer with concrete examples of how to use the DLL in external applications.

A copy of this test program, including fully marked-up C++ source code, can be found on our website at the following URL: www.hyperlabsinc.com/ZTDR.aspx

Section 4: User-facing Functions

The following functions can be accessed by the user at run-time using the ZTDR™ DLL. Each function is presented below with a summary, definition, parameters, return value, and other remarks.

This document presents the functions in required or recommended ***order of call***. Where a function must be called in a specific order, it is noted below.

Section 4.1 - initDevice

Summary

> This function is used to initialize and calibrate the FTDI device used by the TDR. Both the timebase and vertical amplitude are calibrated in sequence. It must be called ***prior to any other function calls***.

Definition

int initDevice (void);

Parameters

No parameters are passed to this function.

Return Value

Returns 1 if successful, else an integer < 0.

Other Notes

This function must be called at instrument startup, prior to any other calls to the DLL. It initializes the TDR, calibrates both the timescale and the vertical amplitude, and prepares the device for data acquisition. No other functions in the DLL will work if ***initDevice*** is not called first.

Section 4.2 - vertCal

Summary

This function is used to calibrate the waveform amplitude to 50 Ω on demand. This function is called by ***initDevice***, so it does not need to be run separately at startup.

Definition

int vertCal (void);

Parameters

No parameters are passed to this function.

Return Value

Returns 1 if successful, else an integer < 0.

Other Notes

This function calibrates the vertical amplitude to 50 Ω and ensures optimal accuracy during data acquisition. It is called as by ***initDevice***, so it is not necessary to run ***vertCal*** immediately after startup. During continuous data acquisition, we recommend calling ***vertCal*** at least every 30 seconds to maintain optimal calibration.

Section 4.3 - setEnviron

Summary

This function sets the environmental variables for the waveform acquisition. It must be called after ***initDevice***, but prior to running any other functions.

Definition

int setEnviron (int x, int y, double start, double end, double k, int rec);

Parameters

The following parameters are passed to this function:

- x Units for the X-axis (0 = m, 1 = ft, 2 = ns)
- y Units for the Y axis (0 = mV, 1 = normalized, 2 = Ohm, 3 = Rho)
- start Start of the acquisition window (0.00 < start < 400.0)
- end End of the acquisition window (end > start; 0.00 < end < 400.0)
- k Dielectric K of the device under test (2.25 for standard coax; 2.05 for Teflon coax)
- rec Number of data points to acquire; ***in version 1.0.1, must be 1024***

Return Value

Returns 1 if successful.

Other Notes

The acquisition environment must be set after ***initDevice***, but prior to running any other functions. It is not necessary to call this function prior to every acquisition, only when an environmental variable is changed.

Section 4.4 - setRefX

Summary

This optional function call sets the horizontal reference (zero) for all subsequent waveform acquisitions. The reference point can either be specified manually or determined automatically by the software. It can be called at any point after ***setEnviron*** has been called.

Definition

int setRefX (double x);

Parameters

The following parameters are passed to this function:

- x Horizontal axis value at which the horizontal reference point (zero) is set (-1.0 = derived automatically, 0.0 = remove offset, x > 0.0 sets to specified value)

Return Value

Returns 1 if successful, else 0.

Other Notes

If the horizontal reference point is to be derived automatically (argument x = -1.0), the system must be in open, i.e. unterminated and not in short. The reference point is set to the point at which the system goes to open, which can be either the output port of the instrument or the end of a reference cable.

Section 4.5 - acquireWaveform

Summary

This function acquires the actual waveform data from the FTDI device, based on the active environmental variables set by ***setEnviron***.

Definition

int acquireWaveform (int numAvg);

Parameters

The following parameters are passed to this function:

- numAvg Number of waveforms to acquire and average together (1 = no averaging)

Return Value

Returns 1 if successful, else 0.

Other Notes

This function must be run for every acquisition. Environmental variables do not need to be re-written prior to every acquisition. To use the acquired data, use ***dumpFile***, ***fetchDataX***, and/or ***fetchDataY***

Section 4.6 - dumpFile

Summary

This function dumps the data acquired by ***acquireWaveform*** into a CSV file for storage and/or processing.

Definition

Int dumpFile (char *filename);

Parameters

The following parameters are passed to this function:

- filename The path and file name, including extension, of the new file to be saved.

Return Value

Returns 1 if successful.

Other Notes

This function dumps the most recent data written to memory by ***acquireWaveform***, in the units set by ***setEnviorn***. This function can only write to a new file; it cannot append to an existing file. If only the filename is given (e.g. “write_data.csv”), the file will be placed in the directory of the executable.

This function dumps a header row, along with the acquired data. The header row stores the important environmental variables (***y units***, ***x units***, ***start***, ***end***, ***k***, ***horizontal reference***). In the subsequent rows, Y values are stored in Column 1, while X values are stored in Column 2.

NOTE: as of Version 1.1.0 of the DLL, the header row stores the shorthand abbreviation of the selected units (e.g. ns, mV, Ohm) instead of the numerical index of each (e.g. 0, 1, 2).

Section 4.7 - fetchDataX

Summary

This function acquires the horizontal (time or distance) value of a single data point currently in memory, in the unit set by ***setEnviornment***.

Definition

double fetchDataX (int idx);

Parameters

The following parameters are passed to this function:

- idx The index of the data point (0 <= idx <= 1023) to be retrieved

Return Value

Returns the X value of the data point, in the selected unit (m, ft, or ns).

Other Notes

This function fetches a value from the most recent data written to memory by ***acquireWaveform***. Used along with fetchDataY in a loop (0 <= idx <= 1023), this function can be used to store every individual data point for post-processing. This function works directly from memory, so it is not necessary to run ***dumpFile*** first.

Section 4.8 - fetchDataY

Summary

This function acquires the vertical (mV, norm, Ohm, Rho) value of a single data point currently in memory, in the unit set by ***setEnviorn***.

Definition

double fetchDataY (int idx);

Parameters

The following parameters are passed to this function:

- idx The index of the data point (0 <= idx <= 1023) to be retrieved

Return Value

Returns the Y value of the data point, in the selected unit (mV, norm, Ohm, or Rho).

Other Notes

This function fetches a value from the most recent data written to memory by ***acquireWaveform***. Used along with fetchDataX in a loop (0 <= idx <= 1023), this function can be used to store every individual data point for post-processing. This function works directly from memory, so it is not necessary to run ***dumpFile*** first.