HYPERLABS
Innovation at Hyperspeed since 1992

HYPERLABS XTDR™

DLL Programmer's Guide (v 2.6.x)

Revision 2.6.0 | www.hyperlabsinc.com

Introduction

These brief reference materials apply to the XTDR™ DLL v2.6.x, compatible with all recent HL22xx and HL52xx TDR instruments purchased from HYPERLABS in 2015 or later.

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 HL22xx and HL52xx TDR functionality outside of the normal Windows® XTDR™ 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.

NOTE: as of Version 2.6, only TDR acquisitions are supported by the DLL. It is not currently possible to use the DLL to acquire data in other modes (TDT, NEXT, FEXT, Insertion Loss, or Return Loss). These features may be added at a later date.

Section 2: Dependencies

The XTDR™ DLL and HL22xx and HL52xx hardware are dependent on the FTDI drivers distributed with XTDR™. When working with the DLL, the following files should all be accessible to the complier:

- FTD2XX.dll
- FTD2XX.h
- FTD2XX.lib
- XTDR_2XX.dll
- XTDR_2XX.h
- XTDR_2XX.lib

Section 3: XTDR™ DLL Test Program

HYPERLABS has also created a XTDR™ 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 marked-up C++ source code, can be found on our website at the following URL: www.hyperlabsinc.com/XTDR.aspx

Section 4: User-facing Functions

The following functions can be accessed by the user at run-time using the XTDR™ 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 - xtdr_init

Summary

This function is used to initialize and calibrate the FTDI device used by the TDR, setting the time base and other vital instrument parameters. It must be called ***prior to any other function calls***.

Definition

int xtdr_init (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, sets the timescale, and prepares the device for data acquisition. No other functions in the DLL will work if ***xtdr_init ()*** is not called first.

Section 4.2 - xtdr_cal

Summary

This function is used to: (1) set/change the instrument acquisition channel; and (2) perform an on-demand calibration to a 50 Ω internal reference line. This function ***is mandatory for instrument functionality***, so it needs to be run at startup and any time the acquisition channel is changed.

NOTE: The amplitude calibration on HL22xx and HL52xx instruments is channel-dependent. For this reason, it is not possible to change the acquisition channel without recalibrating the instrument, nor is it possible to calibrate without specifying a channel.

Definition

int xtdr_cal (int stim, int sampl, int getEdge);

Parameters

The following parameters are passed to this function:

• stim	TDR stimulus channel designation; see *Appendix A* below for stimulus codes
• sampl	TDR sampler channel designation; see *Appendix A* below for sampler codes
• getEdge	Whether to perform TDR edge stabilization (1 = stabilize step to 2 ns; 0 = no stabilization)

Return Value

Returns 1 if successful, else an integer < 0.

Other Notes

It is generally recommended to set ***getEdge == 1*** in most test scenarios. Please contact HYPERLABS if you need help with this parameter.

During continuous data acquisition, we recommend calling ***xtdr_cal*** at least every 60 seconds to maintain optimal amplitude calibration.

Section 4.3 - xtdr_environ

Summary

This function sets the environmental variables for the waveform acquisition. This function is optional. If it is not run, default values (see below) will be used during acquisition.

Definition

int xtdr_environ (int x, int y, double start, double end, double k);

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 < 75.0)
end	End of the acquisition window (end > start; 0.00 <= end < 75.0)
k	Dielectric K of the device under test (2.25 for standard coax; 2.05 for Teflon coax)

Return Value

Returns 1 if successful.

Other Notes

The acquisition environment can be set any time after ***initDevice***. It is not necessary to call this function prior to every acquisition, only when an environmental variable is changed.

If this optional function is not run, the following default values will be used: x = 0 (m), y = 0 (mV), start = 0 (m), end = 2 (m), k = 2.25 (coax)

Section 4.4 - xtdr_zero

Summary

This optional function call sets the horizontal zero reference 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 both ***xtdr_cal*** and ***xtdr_environ*** have been called.

Definition

int xtdr_zero (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 - xtdr_acquire

Summary

This function acquires the actual ***single-ended*** waveform data from the FTDI device, based on the active channel (set by ***xtdr_cal***) and the active environmental variables (set by ***xtdr_environ***, or defaults).

Definition

int xtdr_acquire (int acqNum);

Parameters

The following parameters are passed to this function:

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

Return Value

Returns 1 if successful, else 0.

Other Notes

If differential data is to be acquired, please use the ***xtdr_diff*** function in *Section 4.6* below.

This function is run at every single-ended acquisition. Environmental variables do not need to be re-written prior to every acquisition. To use the acquired data, use ***xtdr_dump***, ***xtdr_getX***, and/or ***xtdr_getY***.

Section 4.6 - xtdr_diff

Summary

This function acquires the actual ***differential*** waveform data from the FTDI device, based on the active channel (set by ***xtdr_cal***) and the active environmental variables (set by ***xtdr_environ***, or defaults).

Definition

int xtdr_diff (int acqNum);

Parameters

The following parameters are passed to this function:

• acqNum	Number of waveforms to acquire and average together (1 = no averaging)

Return Value

Returns 1 if successful, else 0.

Other Notes

If single-ended data is to be acquired, please use the ***xtdr_acquire*** function in *Section 4.5* above.

This function always acquires differential data from the opposite-polarity channels on the same port/channel. For example, if ***Port 1 Channel 1*** is active, differential data is acquired from CH1+ and CH1-.

Differential data is only relevant to output in ***impedance (Ohms)***. If this command is run in mV, Norm, or Rho, the output will not convey useful information.

This function is run at every differential acquisition. Environmental variables do not need to be re-written prior to every acquisition. To use the acquired data, use ***xtdr_dump***, ***xtdr_getX***, and/or ***xtdr_getY***.

Section 4.7 - xtdr_dump

Summary

This function dumps the data acquired by ***xtdr_acquire*** or ***xtdr_diff*** into a CSV file for storage and/or post-processing.

Definition

Int xtdr_dump (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 ***xtdr_acquire*** or ***xtdr_diff***, in the units set by ***xtdr_environ***. 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 (***x units***, ***y units***, ***start***, ***end***, ***k***, ***horizontal zero reference***). In the subsequent rows, X values are stored in Column 1, while Y values are stored in Column 2.

Section 4.8 - xtdr_getX

Summary

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

Definition

double xtdr_getX (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 ***xtdr_acquire*** or ***xtdr_diff***. Used along with xtdr_getY 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 ***xtdr_dump*** first.

Section 4.9 - xtdr_getY

Summary

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

Definition

double xtdr_getY (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 ***xtdr_acquire*** or ***xtdr_diff***. Used along with xtdr_getX 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 ***xtdr_dump*** first.

Appendix A: Stimulus and Sampler Port Codes

On all XTDR™-compatible instruments, stimulus and sampler ports are assigned using numeric values that correspond to pins on the MUX. To acquire data from the correct channel, the correct stimulus and sampler number must be passed to the MUX prior to acquisition using a call to ***xtdr_cal*** (see *Section 4.2* above).

Please use the tables below to determine the port codes of the channel you which to stimulate and sample.

NOTE: some of the ports below are found only on certain instruments. Which ports are assignable depends on the number of channels of the actual device (e.g. HL2202 has 2 channels; HL5208 has 8 channels).

Instrument Channel Label	Stimulus Port Code	Sampler Port Code	Applies To
PORT 1, CH 1+	2	4	2+ channel devices
PORT 1, CH 1-	2	5	2+ channel devices
PORT 1, CH 2+	1	2	8+ channel devices
PORT 1, CH 2-	1	3	8+ channel devices
PORT 1, CH 3+	0	0	12+ channel devices
PORT 1, CH 3-	0	1	12+ channel devices
PORT 1, CH 4+	3	6	16+ channel devices
PORT 1, CH 4-	3	7	16+ channel devices
PORT 1, CH 5+	4	8	20+ channel devices
PORT 1, CH 5-	4	9	20+ channel devices
PORT 2, CH 1+	2	14	4+ channel devices
PORT 2, CH 1-	2	15	4+ channel devices
PORT 2, CH 2+	6	12	8+ channel devices
PORT 2, CH 2-	6	13	8+ channel devices
PORT 2, CH 3+	5	10	12+ channel devices
PORT 2, CH 3-	5	11	12+ channel devices
PORT 2, CH 4+	8	16	16+ channel devices
PORT 2, CH 4-	8	17	16+ channel devices
PORT 2, CH 5+	9	18	20+ channel devices
PORT 2, CH 5-	9	19	20+ channel devices