Optical Backscatter Reflectometer 4200 (Handheld OBR) Software Development Kit (OBR4200 SDK) User Guide

Luna Technologies Application Note and

Addendum to the Optical Backscatter Reflectometer User Guide

Contents

Overview

The OBR4200 Software Development Kit (OBR4200 SDK) or Handheld SDK (HH SDK) is a collection of software routines that allows users of Luna Technologies' OBR4200 to develop custom software utilities targeted for a specific measurement application. By utilizing the SDK software package, OBR4200 users have the ability to write custom graphical user interfaces and data analysis packages using the same source code used to develop the instrument operational software.

The SDK includes sample code in C++ and LabVIEW. The example programs were written and tested using Visual Studio 2005 and LabVIEW 2010.

The example programs call functions contained in DLLs (primarily OBR4000.DLL) to control the OBR4200 and to perform calculations. It may be possible to call these same DLL functions from other programming languages; however C++ and LabVIEW are the only languages currently supported by the SDK.

Getting Started

Copying SDK Files

The SDK consists of a set of directories and files, which should be copied to a directory on your PC. For the sake of clarity, it is recommended that the software be copied to the folder *C:\HHSDK*. They can be placed in an alternate path if desired, but the remainder of these instructions will assume that you have used the default name.

The SDK contains the following folders:
Example – contains C++ example and Visual Studio 2005 project file
include – contains the HHSDK C++ header file
lib – contains the DLLs, .lib, and the .llb which contains the LabVIEW code and examples
Setup – contains the Microsoft Runtime installer and the setup batch file

Runtime Installer

The SDK DLLs require that a specific set of folders and the Microsoft C++ Runtime Library be installed on your computer. To do this, run *setup.bat* from the *HHSDK*\Setup folder. If the computer running the SDK already has the OBR4200 software installed, this step should be unnecessary.

Using the OBR4200 SDK

The easiest way to begin using the SDK is to run the provided example programs. These programs demonstrate the various operations that can be performed with the SDK, and can serve as starting templates for writing your own programs.

LabVIEW Support

Low Level VIs and Example Programs

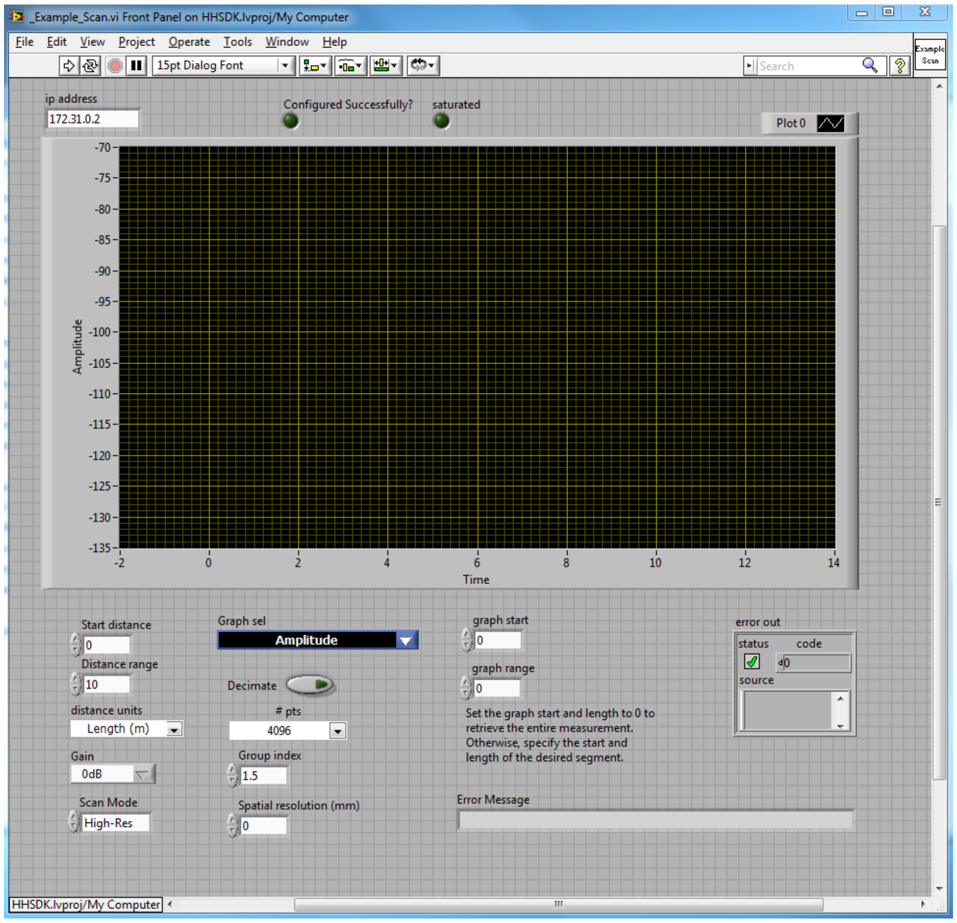
The LabVIEW LLB *C:\HHSDK\lib\HHSDK.llb* contains low level VIs that call the DLLs in the *C:\HHSDK\lib* folder. Most of these VIs are C++ to LabVIEW wrappers and serve as building blocks for writing more complex programs.

The LabVIEW LLB also contains example programs that use the low level VIs to control the OBR4200 and compute measurements. These programs demonstrate how to load configurations, connect to the system, retrieve and set information, acquire measurements, log and load measurements, as well as process and plot data.

This guide will now explain each of the example programs.

_Example_Scan.vi

_Example_Scan.vi shows the steps that must be done to initialize the OBR4200, acquire a measurement, and display result data.


_Example_Scan.vi Front Panel on HHSDK.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Dialog Font
Search
Example Scan
Plot 0
Amplitude
Time
HHSDK.lvproj/My Computer

ip address: 172.31.0.2
Configured Successfully?
saturated
Start distance: 0
Distance range: 10
distance units: Length (m)
Gain: 0dB
Scan Mode: High-Res
Graph sel: Amplitude
Decimate
# pts: 4096
Group index: 1.5
Spatial resolution (mm): 0
graph start: 0
graph range: 0
Set the graph start and length to 0 to retrieve the entire measurement. Otherwise, specify the start and length of the desired segment.
error out
status code: 0
source
Error Message

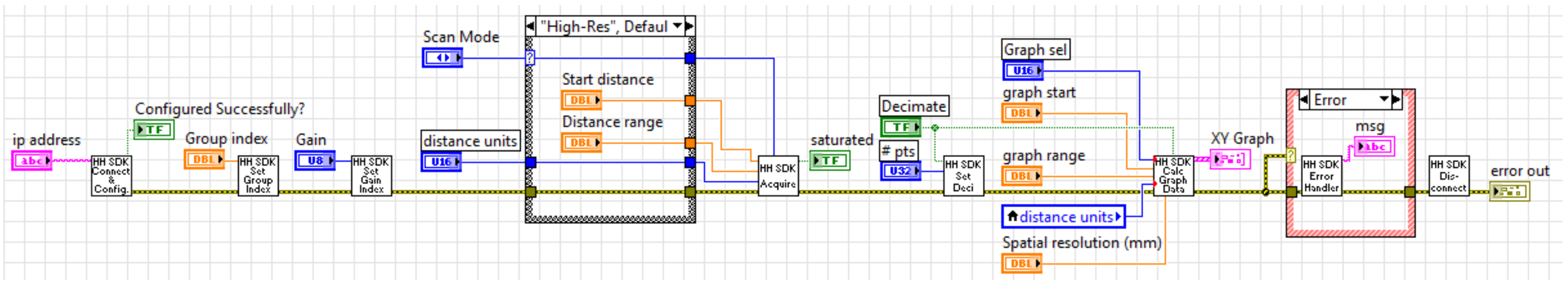


Figure 1 _Example_Scan.vi

1. *HHSDK_ConnectAndConfigure.vi* connects to and initializes the OBR. This should be done at the beginning of any HH SDK program.

2. *HHSDK_SetGroupIndex.vi* sets the group index.

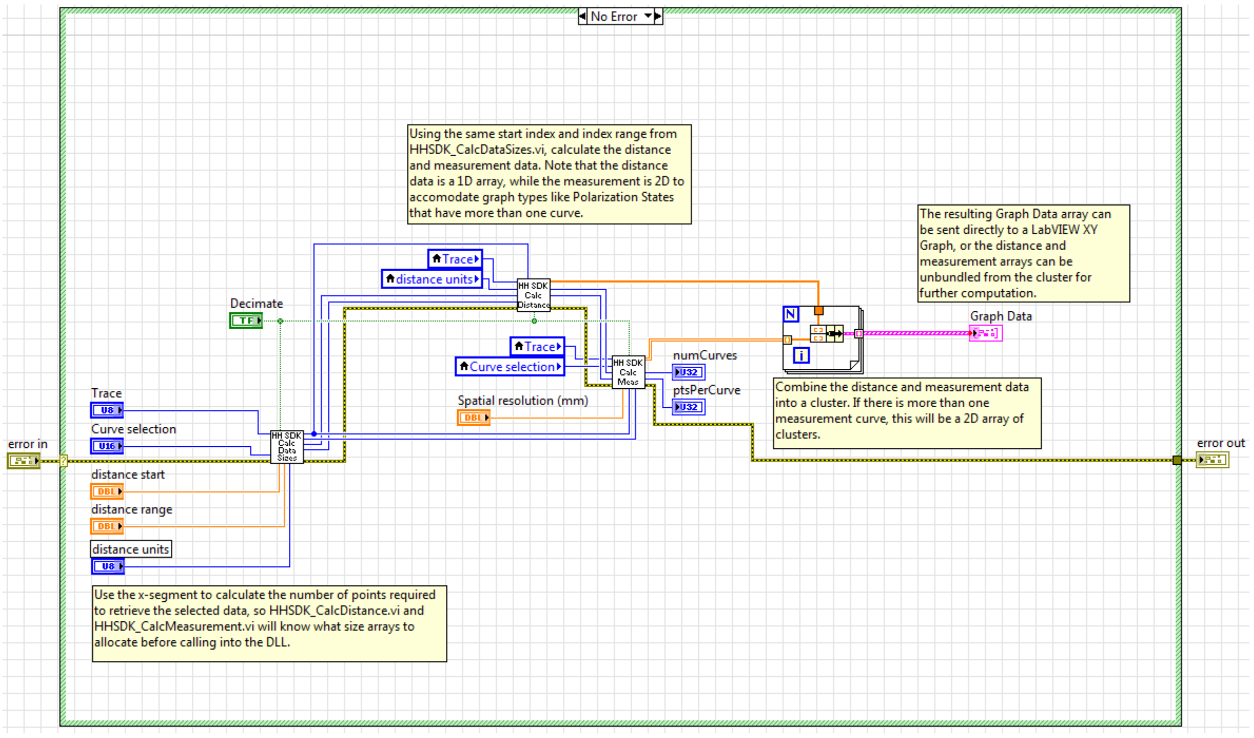
3. *HHSDK_SetGainIndex.vi* sets the gain index.
    - It is important to note the value set here is not the gain in dB, but the gain index. A set of distinct gain values (typically 0, 6, 12, 18, and 24 dB) are available when taking an OBR measurement. These gain values are selected by setting the gain index to 0, 1, 2, 3, or 4.

4. Before calling Acquire, a case statement sets up the requirements for either a Low Resolution or High Resolution Scan. A low resolution scan acquires low resolution data for the entire distance range that the OBR is capable of. A high resolution scan acquires detailed data for a smaller distance range.
    - The Low Resolution Scan calls HHSDK_GetmaxRange.vi to determine the maximum distance range.
    - The start distance for Low Resolution should always be 0.
    - The High Resolution Scan uses the user-specified start distance and range.

5. *HHSDK_Acquire.vi* acquires a measurement.
    - Calls two VIs:
        - *HHSDK_ConvertUnits.vi* converts the start distance and range to ns.
        - *HHSDK_AcqMeas.vi* acquires a measurement.

6. *HHSDK_SetDecimation.vi* controls whether *HHSDK_CalcGraphData.vi* returns all of the data points for the selected segment or returns a decimated data array.
    - If the Decimate control is set to True, the software calls HHSDK_SetNumDeciPts.vi to set the desired number of points.

7. *HHSDK_CalcGraphData.vi* computes the measurement data and outputs it to the XY Graph.



Using the same start index and index range from HHSDK_CalcDataSizes.vi, calculate the distance and measurement data. Note that the distance data is a 1D array, while the measurement is 2D to accomodate graph types like Polarization States that have more than one curve.

The resulting Graph Data array can be sent directly to a LabVIEW XY Graph, or the distance and measurement arrays can be unbundled from the cluster for further computation.

Combine the distance and measurement data into a cluster. If there is more than one measurement curve, this will be a 2D array of clusters.

Use the x-segment to calculate the number of points required to retrieve the selected data, so HHSDK_CalcDistance.vi and HHSDK_CalcMeasurement.vi will know what size arrays to allocate before calling into the DLL.

Figure 2 HHSDK_CalcGraphData.vi

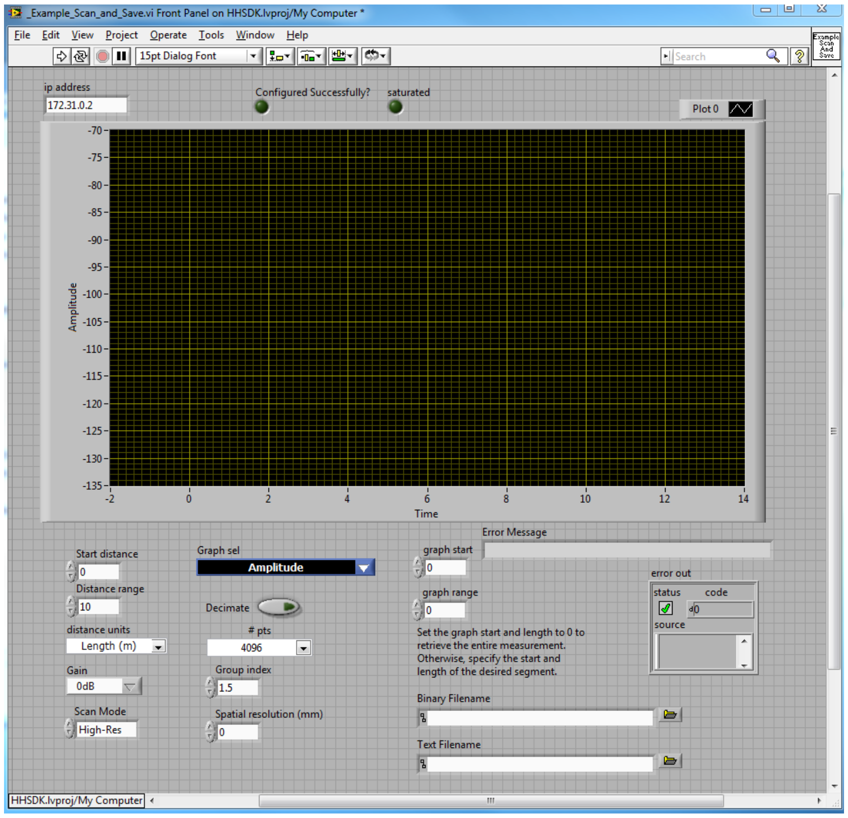
- It calls three other VIs:
  - *HHSDK_CalcDataSizes.vi*
    - Calculates the sizes of the data sets that will be returned for the specified graph type.
  - *HHSDK_CalcDistance.vi*
    - Calculates the distance data from the requested region.
  - *HHSDK_CalcMeasurment.vi*
    - Calculates the measurement data for the requested region.

8. *HHSDK_GetErrorMsg.vi* is called if an error occurred during the setup or acquisition.
   - This VI calls the OBR4200 DLLs to determine what error occurred and displays the appropriate message.

9. *HHSDK_Disconnect.vi* disconnects the OBR4200 from the computer. This should be done at the end of any HH SDK program.

_Example_Scan_and_Save.vi

The purpose of this example VI is to demonstrate the save features that the Handheld SDK provides. It uses the same setup and acquire code as *_Example_Scan.vi*.



ip address: 172.31.0.2

Start distance: 0

Distance range: 10

distance units: Length (m)

Gain: 0dB

Scan Mode: High-Res

Graph sel: Amplitude

Decimate

pts: 4096

Group index: 1.5

Spatial resolution (mm): 0

graph start: 0

graph range: 0

Set the graph start and length to 0 to retrieve the entire measurement. Otherwise, specify the start and length of the desired segment.

Error Message

error out: status ✓ code 0 source

Binary Filename

Text Filename

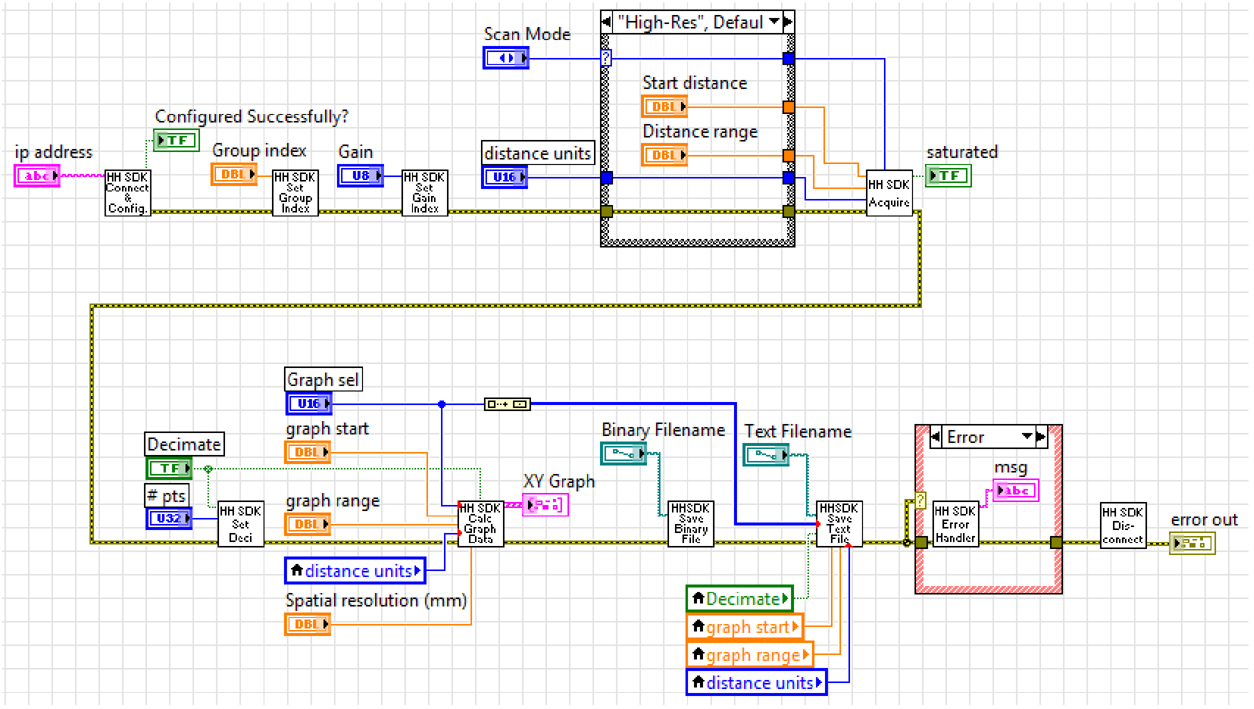


Figure 3 _Example_Scan_and_Save.vi

1. *HHSDK_SaveBinaryFile.vi* saves the chosen trace as a binary file to the provided path.

2. *HHSDK_SaveTextFile.vi* saves the chosen trace as a text file to the provided path.

_Example_AcquireReference.vi

The purpose of this example VI is to show how to acquire a reference and acquire information about the current reference.

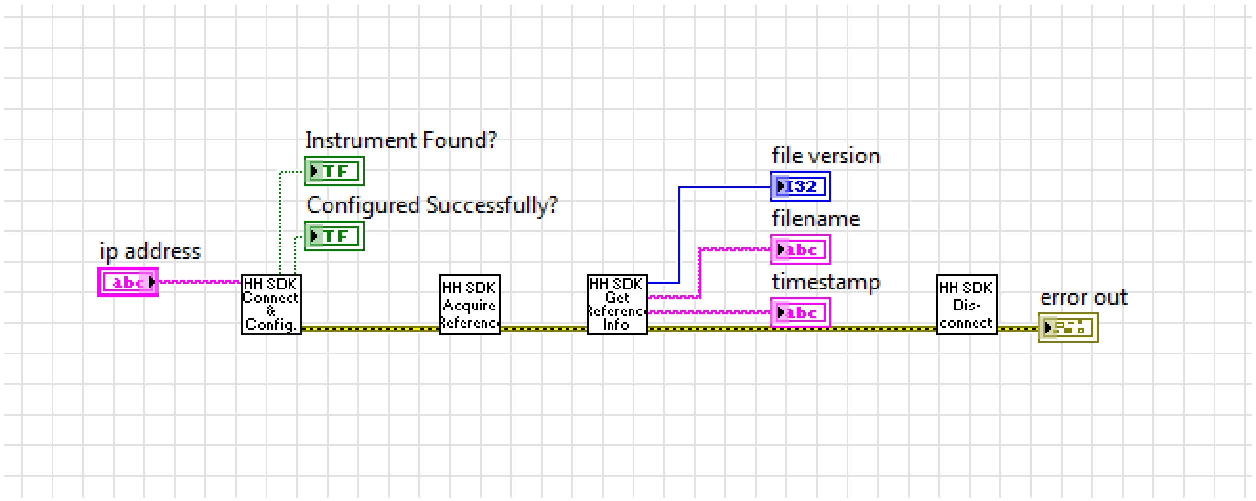


Figure 4 _Example_AcquireReference.vi

1. *HHSDK_AcquireReference.vi* acquires a reference scan.

2. *HHSDK_GetReferenceInfo.vi* acquires information about the current reference.

_Example_SetIPAddress.vi

_Example_SetIPAddress.vi shows how to change the current IP address.

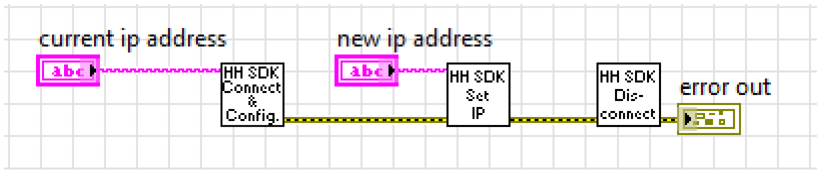


Figure 5 _Example_SetIPAddress.vi

1. *HHSDK_SetIPAddress.vi* changes the current IP address. The OBR 4200 must be power cycled before this change will take effect.
2. Whenever the IP address is changed, the new value is written to the file C:\ProgramData\Luna Technologies\OBR4200\ipaddress.txt.

C/C++ Support

There is one C/C++ header file included with the SDK. This file is all that is required to begin compiling and working with the library.

- *HHSDK.h* – the main header file that contains all of the prototypes of the functions that are specific to the OBR4200 product. This header file also contains all enumeration types and structure definitions to allow for easy integration of the SDK code into another C/C++ project.
    - To see function definitions, check below section labeled "C/C++ Function Definitions"

In addition to the header file mentioned above, a sample workspace named "HHSDK_Example" is also bundled with the SDK source code. This workspace was created and compiled with Microsoft Visual C++ v8.0 (2005). It can be used as the model for creating new projects that are to be used to call functions within the included DLL.

The main source file in the "HHSDK_Example" project contains a single function that shows how one would call the functions in the OBR4000 DLL to perform various actions. It begins by calling the *HHSDK_ConnectAndConfigure* function to initialize the hardware and allocate the necessary memory. If this configuration is successful, the proper initialization can be performed that will allow a measurement to be acquired with the hardware. Upon obtaining measurement data, the appropriate parameters can be calculated to allow graphical display of the data.

C/C++ Function Definitions

HHSDK_CAPI INT32 HHSDK_AcqMeas (UINT8 trace, *FLOAT64* start_ns, *FLOAT64* length_ns, *UINT8* scanMode, *UINT8 ** saturated*)*

HHSDK_AcqMeas

- Acquire a measurement

- Input:
  - UINT8 trace - trace in which to store the measurement
  - FLOAT64 start_ns - start location of measurement in units of ns
  - FLOAT64 length_ns - length of the measurement in units of ns
  - UINT8 scanMode - 0: LowRes 1: HighRes
- Outputs:
  - UINT8 *saturated - if ADCs were saturated due to excessive gain (0/1)

Returns:
    error_status_code

HHSDK_CAPI INT32 HHSDK_AcquireReference ()

HHSDK_AcquireReference

- Acquire a reference

Returns:
    error_status_code

HHSDK_CAPI INT32 HHSDK_CalcDataSizes (UINT8 trace, *UINT16* graphType, *UINT8* decimate, *FLOAT64* start, *FLOAT64* length, *UINT8* units, *UINT32 ** numCurves, *UINT32 ** ptsPerCurve, *UINT32 ** startIndex, *UINT32 ** indexRange*)*

HHSDK_CalcDataSizes

- Calculate the sizes of the data sets that will be returned for the specified graph type.

- Outputs startIndex and indexRange will be used to specify the desired data segment when calling HHSDK_CalcMeasurement and HHSDK_CalcDistance.
- Outputs ptsPerCurve and numCurves should be used to allocate arrays before calling HHSDK_CalcMeasurement and HHSDK_CalcDistance.
- The array passed to HHSDK_CalcDistance should hold at least ptsPerCurve elements.
- The array passed to HHSDK_CalcMeasurement should hold at least ptsPerCurve * numCurves elements.

- Inputs:
  - UINT8 trace - trace number
  - UINT16 graphType - graph type selection
  - UINT8 decimate - 0 to calculate the size of a decimated data set, 1 for the full data set
  - FLOAT64 start - start of the segment to calculate the size of
  - FLOAT64 length - length of segment
  - UINT8 units - units of start and length
- Outputs:
  - UINT32 *numCurves - number of curves the data set will contain
  - UINT32 *ptsPerCurve - number of points in each curve
  - UINT32 *startIndex - starting index of the segment
  - UINT32 *indexRange - length of the segment in index points

Returns:

error_status_code

HHSDK_CAPI INT32 HHSDK_CalcDistance (UINT8 **trace**, *UINT32* **startIndex**, *UINT32* **indexRange**, *UINT8* **decimate**, *UINT8* **units**, *FLOAT64 ** **distance**, *UINT32 ** **arrSize***)*

HHSDK_CalcDistance

- Calculates the distance data for the requested region.
- The parameters startIndex and indexRange should be calculated by calling HHSDK_CalcDataSizes with the desired distance start and range.
- The calling function must allocate the distance array hold at least indexRange elements before calling HHSDK_CalcDistance.

- Inputs:
  - UINT8 trace - trace number

  - UINT32 startIndex - start index of region
  - UINT32 indexRange - number of index points in region
  - UINT8 decimate - 1 to return decimated data, 0 to return the full data set
  - UINT8 units - desired units
- Outputs:
  - FLOAT64 *distance - array of distance points
  - UINT32 *arrSize - number of points in distance array (should be the same as ptsPerCurve from HHSDK_CalcDataSizes)

Returns:

error_status_code

HHSDK_CAPI INT32 HHSDK_CalcMeasurement (UINT8* trace, *UINT16* graphType, *UINT32* startIndex, *UINT32* indexRange, *UINT8* decimate, *FLOAT64* spatialRes, *FLOAT64 ** measurement, *UINT32 ** numCurves, *UINT32 ** ptsPerCurve*)

HHSDK_CalcMeasurement

- Calculates the measurement data for the requested region.
- The parameters startIndex and indexRange should be calculated by calling HHSDK_CalcDataSizes with the desired distance start and range.
- The calling function must allocate the measurement array (using the sizes from HHSDK_CalcDataSizes) before calling HHSDK_CalcMeasurement.

- Inputs:
  - UINT8 trace - trace number
  - UINT16 graphType - graph type
  - UINT32 startIndex - start index of region
  - UINT32 indexRange - number of index points in region
  - UINT8 decimate - 1 to return decimated data, 0 to return the full data set
  - FLOAT64 spatialRes - spatial resolution in mm
- Outputs
  - FLOAT64 *measurement - array of measurement points
  - UINT32 *numCurves - number of curves the data set will contain
  - UINT32 *ptsPerCurve - number of points in each curve

Returns:

error_status_code

HHSDK_CAPI INT32 HHSDK_CoerceDistance (FLOAT64 **start_ns,** *FLOAT64* **length_ns,** *FLOAT64 ** **coerced_start_ns,** *FLOAT64 ** **coerced_length_ns***)*

HHSDK_CoerceDistance

- Coerce a desired distance range into an exact range that can be acquired.

- Inputs:
  - FLOAT64 start_ns - desired start location in ns
  - FLOAT64 length_ns - desired length in ns
- Outputs:
  - FLOAT64 *coerced_start_ns - coerced start location
  - FLOAT64 *coerced_length_ns - coerced length

Returns:

error_status_code

*HHSDK_CAPI INT32 HHSDK_ConnectAndConfigure (const char ** **ipaddr,** *UINT8 ** **numTraces,** *UINT8 ** **calDataLoaded***)*

HHSDK_ConnectAndConfigure

- Connects and configures the hardware
- Loads the configuration and initializes memory/software parameters

- Inputs:
  - const char *ipaddr - ip address
- Outputs:
  - UINT8 *numTraces - number of traces
  - UINT8 *calDataLoaded - calibration data loaded successfully?

Returns:

error_status_code

HHSDK_CAPI INT32 HHSDK_ConvertUnits (FLOAT64 **val,** *UINT8* **unitsIn,** *UINT8* **unitsOut,** *FLOAT64 ** **convertedVal***)*

HHSDK_ConvertUnits

- Convert value loc from unitsIn to unitsOut
- Possible unit values are:
  - 0 : ns, 1 : m, 2 : ft, 3 : in
- Inputs:
  - FLOAT64 val - value to convert
  - UINT8 unitsIn - units of input val
  - UINT8 unitsOut - units to convert to
- Outputs:
  - FLOAT64 convertedVal - input val converted to units unitsOut

Returns:
error_status_code

HHSDK_CAPI INT32 HHSDK_Disconnect ()

HHSDK_Disconnect

- Close the TCP interface and disconnect from the remote device.
- This function should be called at the end of any HHSDK program.

Returns:
error_status_code

*HHSDK_CAPI INT32 HHSDK_GetDecimationSize (UINT32 ** **numPts***)*

HHSDK_GetDecimationSize

- Gets the number of points that will be returned from HHSDK_CalcMeasurement and HHSDK_CalcDistance when the decimate parameter is set to 1.\

- Outputs:
  - UINT32 *numPts - number of points to return in data arrays

Returns:
    error_status_code

*HHSDK_CAPI INT32 HHSDK_GetErrorMsg (char ** **msg,** *UINT32* **msgSize***)*

HHSDK_GetErrorMsg

- Retrieves the most recent error message

- Inputs:
  - char *msg - error
- Outputs:
  - char msgSize - max size of msg string

Returns:
    error_status_code

*HHSDK_CAPI INT32 HHSDK_GetFreqDomainWindowFlag (UINT8 ** **val***)*

HHSDK_GetFreqDomainWindowFlag

- Gets the value of the frequency domain window flag

- Outputs:
  - UINT8 *val - 0 : not applying frequency domain window, 1 : applying window

Returns:
    error_status_code

HHSDK_CAPI INT32 HHSDK_GetGraphProperties (UINT16 **graphType,** *char ** **graphName,** *UINT32 ** **numCurves,** *char ** **curveNames,** *char ** **units***)*

HHSDK_GetGraphProperties

- Retrieves the labels associated with a given graph type

- Inputs:
  - o UINT16 graphType - graph type selection
- Outputs:
  - o char *graphName - name of this graph type
  - o UINT32 *numCurves - number of curves plotted by this graph type
  - o char *curveNames - name(s) of the curve(s) plotted by this graph type
  - o char *units - measurement units associated with this graph type

Returns:
error_status_code

*HHSDK_CAPI INT32 HHSDK_GetGroupIndex (FLOAT64 ** **grpIndex*)

HHSDK_GetGroupIndex

- Get the current group index

- Outputs:
  - o FLOAT64 *grpIndex - group index

Returns:
error_status_code

HHSDK_CAPI INT32 HHSDK_GetMaxRange (UINT8 **units,** *FLOAT64 ** **maxRange)**

HHSDK_GetMaxRange

- Get the maximum length for HHSDK scans

- Inputs:
  - o UINT8 units - units for maxRange
- Outputs:
  - o FLOAT64 *maxRange - maximum distance range

Returns:

error_status_code

*HHSDK_CAPI INT32 HHSDK_GetReferenceInfo (INT32 ** **version,** ***char** ** **filename,** ***char** ** **timestamp***)*

HHSDK_GetReferenceInfo

- Get details about the current reference
- Outputs:
    - INT32 *version - file format version
    - char *filename - name of reference file
    - char *timestamp - time stamped when reference was acquired

Returns:

error_status_code

*HHSDK_CAPI INT32 HHSDK_GetSpatialResolutionFilterFlag (UINT8 ** **value***)*

HHSDK_GetSpatialResolutionFilterFlag

- Gets the spatial resolution filter flag setting
- Outputs:
    - UINT8 *value - 0 : filter off, 1 : filter on

Returns:

error_status_code

*HHSDK_CAPI INT32 HHSDK_GetSystemStatus (UINT8 ** **brdTempError,** ***UINT8** ** **lzrTempError***)*

HHSDK_GetSystemStatus

- Retrieves system status.

- Outputs:
  - UINT8 *brdTempError - flag for board over-temperature
  - UINT8 *lzrTempError - flag for laser over-temperature

Returns:
    error code

*HHSDK_CAPI INT32 HHSDK_GetVersionInfo (char ** **swRelInfo,** *char ** **identification,** *char ** **FPGAFwVer,** *char ** **processorFwVer,** *char ** **CPLDFwVer,** *INT32* **maxSize***)*

HHSDK_GetVersionInfo

- Retrives version information

- Outputs:
  - char *swRelInfo - Software Release Information
  - char *identification - Identification
  - char *FPGAFwVer - FPGA Firmware Version
  - char *processorFwVer - Processor Firmware Version
  - char *CPLDFwVer - CPLD Firmware Version
  - INT32 maxSize - Max size of the string outputs String outputs should be allocated to maxSize before calling HHSDK_GetVersionInfo.

Returns:
    error_status_code

HHSDK_CAPI INT32 HHSDK_LoadBinaryFile (UINT8 **trace,** *char ** **filepath***)*

HHSDK_LoadBinaryFile

- Load a binary measurement file into the given trace

- Inputs:
  - UINT8 trace - trace number
  - char *filename - file to load

Returns:
    error_status_code

HHSDK_CAPI INT32 HHSDK_ReloadReference ()

HHSDK_ReloadReference

- Reload the current reference

Returns:
error_status_code

HHSDK_CAPI INT32 HHSDK_RemoveCurrentReference ()

HHSDK_RemoveCurrentReference

- Remove the current reference

Returns:
error_status_code

HHSDK_CAPI INT32 HHSDK_SaveBinaryFile (UINT8 trace, *char ** filepath*)*

HHSDK_SaveBinaryFile

- Saves the binary measurement file held in the given trace

- Inputs:
  - UINT8 trace - trace number
  - char *filename - file to save

Returns:
error_status_code

HHSDK_CAPI INT32 HHSDK_SaveBinarySegment (UINT8 trace, *char ** filepath, *FLOAT64* start, *FLOAT64* length, *UINT8* units*)*

HHSDK_SaveBinarySegment

- Saves a segment of the binary measurement file held in the given trace

- Inputs:
    - UINT8 trace - trace number
    - char *filename - file to save
    - FLOAT64 start - start of segment
    - FLOAT64 length - length of segment
    - UINT8 units - units of start and length (0 : ns, 1 : m, 2 : ft, 3 : in)

Returns:
error_status_code

HHSDK_CAPI INT32 HHSDK_SaveTextFile (UINT8 trace, *char ** filename, *UINT16 ** graphTypes, *UINT16* numGraphTypes, *UINT8* decimate, *FLOAT64* spatRes, *FLOAT64* start, *FLOAT64* length, *UINT8* units*)*

HHSDK_SaveTextFile

- Saves a text file containing measurement and distance data for one or more graph types

- Inputs:
    - UINT8 trace - trace number
    - char *filename - name of output file
    - UINT16 *graphTypes - array of graph types to save
    - UINT16 numGraphTypes - number of graph types in graphTypes array
    - UINT8 decimate - 1 to decimate data, 0 to write full resolution data
    - FLOAT64 spatRes - spatial resolution in mm
    - FLOAT64 start - start distance
    - FLOAT64 length - distance length
    - UINT8 units - distance units (0 : ns, 1 : m, 2 : ft, 3 : in)

Returns:
error_status_code

HHSDK_CAPI INT32 HHSDK_SetDecimationSize (UINT32 numPts*)*

HHSDK_SetDecimationSize

- Sets the maximum number of points that will be returned from HHSDK_CalcMeasurement and HHSDK_CalcDistance when the decimate parameter is set to 1.

- Inputs:
  - numPts - number of points to return in data arrays

Returns:
error_status_code

HHSDK_CAPI INT32 HHSDK_SetFreqDomainWindowFlag (UINT8 val*)*

HHSDK_SetFreqDomainWindowFlag

- Gets the value of the frequency domain window flag

- Inputs:
  - UINT8 val - 0 : do not apply frequency domain window, 1 : apply window

Returns:
error_status_code

HHSDK_CAPI INT32 HHSDK_SetGainIndex (UINT8 newGainIndex, *UINT8 ** prevGainIndex*)*

HHSDK_SetGainIndex

- Sets the gain that will be used when taking measuremts
- This function sets the gain index, not the actual gain in dB
- Possible values are:
  - : 0 dB, 1 : 6 dB, 2 : 12 dB, 3 : 18 db, 4 : 24 dB, 5 : 30 dB

- Inputs:
  - UINT8 newGainIndex - new gain index to use

- Outputs:
    - UINT8 *prevGainIndex - previous gain index

Returns:
    error_status_code

HHSDK_CAPI INT32 HHSDK_SetGroupIndex (FLOAT64 grpIndex*)*

HHSDK_SetGroupIndex

- Sets the group index

- Inputs:
    - FLOAT64 grpIndex - new group index

Returns:
    error_status_code

*HHSDK_CAPI INT32 HHSDK_SetIPAddress (char ** ipaddr*)*

HHSDK_SetIPAddress

- Changes the IP address that the OBR4200 will use.
- This change does not take effect until the OBR4200 is power cycled.

- Inputs:
    - char *ipaddr - new IP address

Returns:
    error_status_code

HHSDK_CAPI INT32 HHSDK_SetSpatialResolutionFilterFlag (UINT8 value*)*

HHSDK_SetSpatialResolutionFilterFlag

- Turns the spatial resolution filter flag on or off

- Inputs:
    - UINT8 value - 0 : filter off, 1 : filter on

Returns:

error_status_code

HHSDK_CAPI INT32 HHSDK_SetZeroLengthLocation (FLOAT64 location*, UINT8* units*)*

HHSDK_SetZeroLengthLocation

- Sets the zero length location

- Inputs:
    - FLOAT64 location - zero length location
    - UINT8 units - units of location (0 : ns, 1 : m, 2 : ft, 3 : in)

Returns:

error_status_code

LabVIEW is a registered trademark of National Instruments Inc.
Visual C++ is a registered trademark of Microsoft Corporation.

Product Support Contact Information

If you should have any problems with or questions about the information contained in this document, please do not hesitate to contact our technical support staff via one of the following methods:

Headquarters:	3157 State Street Blacksburg, VA 24060
Main Phone:	1.540.961.5190
Toll-Free Support:	1.866.586.2682
Fax:	1.540.961.5191
Email:	solutions@lunainc.com
Website:	www.lunainc.com