Integrating a Thorlabs CCS Spectrometer in MATLAB

Introduction:

Thorlabs CCS spectrometers can be controlled in MATLAB using the provided driver DLLs. These can be loaded into MATLAB using the “loadlibrary” command. This example uses the 64-bit version of the CCS drivers. If you want to use the 32-bit versions, please make sure to change the file paths and names accordingly.

The driver documentation files are saved to this folder during installation: C:\Program Files\IVI Foundation\VISA\Win64\TLCCS\Manual

Programming:

1) **Preparing “visatype.h”:**

At first, comment out the “__fastcall” and “signed” calling conventions in the header file “visatype.h” in this folder: C:\Program Files\IVI Foundation\VISA\Win64\Include

```
/*---------------------------------------------------------------------------*/
/* Distributed by IVI Foundation Inc.                                        */
/*                                                                           */
/* Do not modify the contents of this file.                                  */
/*---------------------------------------------------------------------------*/
/*                                                                           */
/* Title   : VISATYPE.H                                                      */
/* Date    : 05-12-2009                                                      */
/* Purpose : Fundamental VISA data types and macro definitions               */
/*                                                                           */
/*---------------------------------------------------------------------------*/
```

```
#ifndef __VISATYPE_HEADER__
#define __VISATYPE_HEADER__

#if defined(_WIN64)
#define _VI_FAR
#define _VI_FUNC            //__fastcall
#define _VI_FUNCC           //__fastcall
#define _VI_FUNCH           //__fastcall
#define _VI_SIGNED          //signed
#elif (defined(WIN32) || defined(_WIN32) || defined(__WIN32__) || defined(__NT__)) &&
!defined(_NI_mswin16_)
#define _VI_FAR
.
.
.
```

If these conventions aren't commented out, they normally cause error messages when you try to run the MATLAB code.

2) Set instrument ID:

The second step is to change the instrument ID in the MATLAB code to the ID of your device. The instrument ID looks like this:

```
USB0::0x1313::<Type-ID>::<Serial Number>::0::RAW
```

The type-ID depends on the spectrometer:

• CCS100 Compact Spectrometer:	0x8081
• CCS125 Special Spectrometer:	0x8083
• CCS150 UV Spectrometer:	0x8085
• CCS175 NIR Spectrometer:	0x8087
• CCS200 UV-NIR Spectrometer:	0x8089

The serial number is written on the backside of the spectrometer.

For the CCS175 with serial number M00234008 the instrument ID looks e.g. like this:

```
USB0::0x1313::0x8087::M00234008::0::RAW
```

This step is necessary to make sure that the right device is accessed.

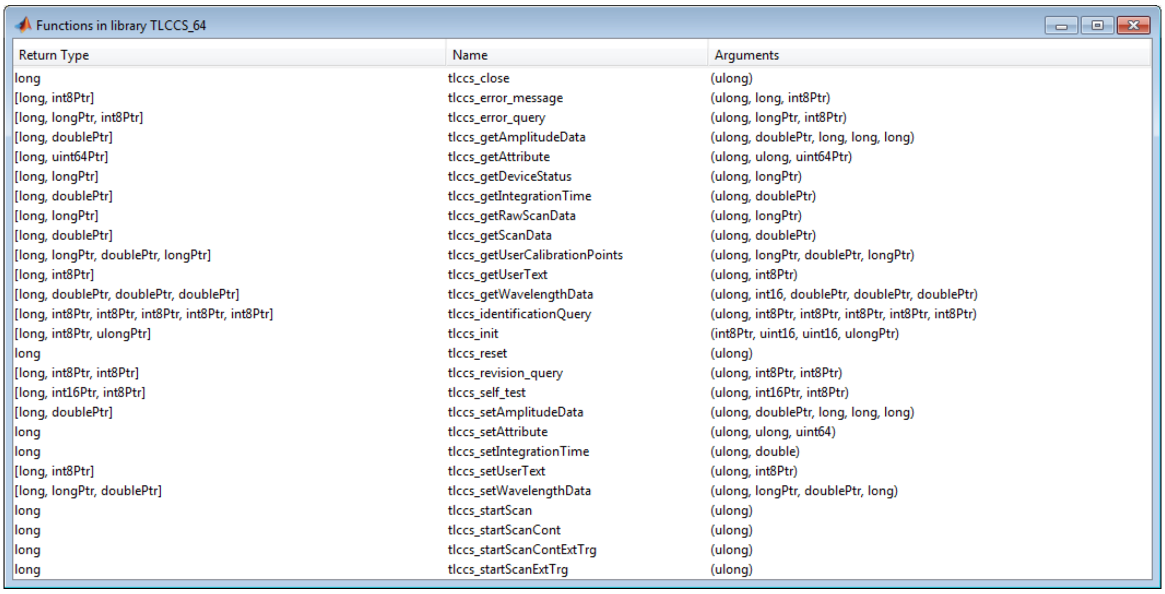
3) Sample code:

This sample code can be copied into MATLAB and can be executed once the changes in 1) and 2) are made.

```
clc;
clear;
disp('Start spectrometer.');

%   Loading the dll and header file into MATLAB
libname='C:\Program Files\IVI Foundation\VISA\Win64\Bin\TLCCS_64.dll';
hfile='C:\Program Files\IVI Foundation\VISA\Win64\Include\TLCCS.h';
loadlibrary(libname,hfile,'includepath','C:\Program Files\IVI
Foundation\VISA\Win64\Include\', 'includepath', 'C:\Program Files\IVI
Foundation\VISA\Win64\Lib_x64\');
disp('Library loaded.');

%   Displays the functions in the library
%   Also gives the data types used in a command
%   - Not necessary for normal use -
libfunctionsview 'TLCCS_64';

%   Some dll functions use pointers
%   The 'libpointer' command has to be used in MATLAB for this

%   !!! Change this instrument ID to the ID of your device !!!
%
%   Type-ID:
%   0x8081   // CCS100 Compact Spectrometer
%   0x8083   // CCS125 Special Spectrometer
%   0x8085   // CCS150 UV Spectrometer
%   0x8087   // CCS175 NIR Spectrometer
%   0x8089   // CCS200 UV-NIR Spectrometer
%
%   'USB0::0x1313::<Type-ID>::<Serial Number>::0::RAW'

%   Initialize the spectrometer
res=libpointer('int8Ptr',int8('USB0::0x1313::0x8087::M00234008::0::RAW'));
hdl=libpointer('ulongPtr',0);
[a,b,c]=calllib('TLCCS_64', 'tlccs_init', res, 0, 0, hdl);
disp(['Initialize device (0 = correct, rest = error): ', num2str(a)]);

%   Open figure window
figure;

for i=1:10

%   Set integration time, measure spectrum and get data
inttime=0.1;
calllib('TLCCS_64','tlccs_setIntegrationTime',hdl.value,inttime);
calllib('TLCCS_64', 'tlccs_startScan', hdl.value);
%pause(1);
specdata=libpointer('doublePtr',double(1:3648));
calllib('TLCCS_64','tlccs_getScanData', hdl.value, specdata);
wldata=libpointer('doublePtr',double(1:3648));
calllib('TLCCS_64','tlccs_getWavelengthData', hdl.value, 0, wldata, 0, 0);

%   Display spectrum
```

```
pause(0.001);
plot(wldata.value,specdata.value);
title(['Spectrum ', num2str(i), ' (Integration time: ', num2str(inttime), '
sec)']);
xlabel('Wavelength [nm]');
ylabel('Counts [a.u.]');

end;

%   Close spectrometer connection, unload library
calllib('TLCCS_64','tlccs_close', hdl.value);
unloadlibrary 'TLCCS_64';
```

The command “`libfunctionsview 'TLCCS_64'`” will open a window like this:



Functions in library TLCCS_64

Return Type	Name	Arguments
long	tlccs_close	(ulong)
[long, int8Ptr]	tlccs_error_message	(ulong, long, int8Ptr)
[long, longPtr, int8Ptr]	tlccs_error_query	(ulong, longPtr, int8Ptr)
[long, doublePtr]	tlccs_getAmplitudeData	(ulong, doublePtr, long, long, long)
[long, uint64Ptr]	tlccs_getAttribute	(ulong, ulong, uint64Ptr)
[long, longPtr]	tlccs_getDeviceStatus	(ulong, longPtr)
[long, doublePtr]	tlccs_getIntegrationTime	(ulong, doublePtr)
[long, longPtr]	tlccs_getRawScanData	(ulong, longPtr)
[long, doublePtr]	tlccs_getScanData	(ulong, doublePtr)
[long, longPtr, doublePtr, longPtr]	tlccs_getUserCalibrationPoints	(ulong, longPtr, doublePtr, longPtr)
[long, int8Ptr]	tlccs_getUserText	(ulong, int8Ptr)
[long, doublePtr, doublePtr, doublePtr]	tlccs_getWavelengthData	(ulong, int16, doublePtr, doublePtr, doublePtr)
[long, int8Ptr, int8Ptr, int8Ptr, int8Ptr, int8Ptr]	tlccs_identificationQuery	(ulong, int8Ptr, int8Ptr, int8Ptr, int8Ptr, int8Ptr)
[long, int8Ptr, ulongPtr]	tlccs_init	(int8Ptr, uint16, uint16, ulongPtr)
long	tlccs_reset	(ulong)
[long, int8Ptr, int8Ptr]	tlccs_revision_query	(ulong, int8Ptr, int8Ptr)
[long, int16Ptr, int8Ptr]	tlccs_self_test	(ulong, int16Ptr, int8Ptr)
[long, doublePtr]	tlccs_setAmplitudeData	(ulong, doublePtr, long, long, long)
long	tlccs_setAttribute	(ulong, ulong, uint64)
long	tlccs_setIntegrationTime	(ulong, double)
[long, int8Ptr]	tlccs_setUserText	(ulong, int8Ptr)
[long, longPtr, doublePtr]	tlccs_setWavelengthData	(ulong, longPtr, doublePtr, long)
long	tlccs_startScan	(ulong)
long	tlccs_startScanCont	(ulong)
long	tlccs_startScanContExtTrg	(ulong)
long	tlccs_startScanExtTrg	(ulong)

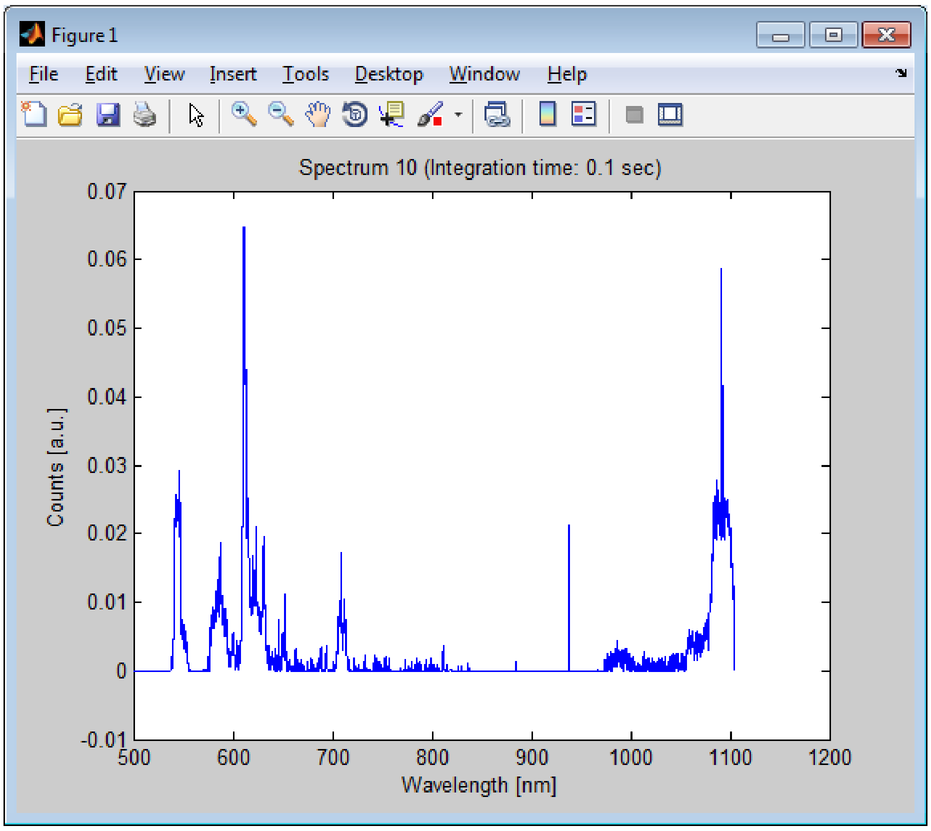
It shows the commands in the driver dll file which can be used with the CCS spectrometer. It is useful for the programming in MATLAB because it also shows which data types are expected as input / output variables by MATLAB.

Once the programming is finished, this window will be unnecessary in most cases and can be commented out.

The measurement part of the program records 10 spectra and displays them following these steps:

- Set the integration time (0.1 sec = 100 ms, in this case)
- Start one scan
- Get the scan data
- Get the wavelength data (the corresponding wavelength for each pixel)
- Plot the spectrum (scan and wavelength data) to the figure window

You can see an example of the figure window below:



For further assistance please contact us on: europe@thorlabs.com / +49 (0) 8131-5956-2