MATLAB support for the AvaSpec

The DLL package now includes support for MATLAB. Both a 32 bit and a 64 bit support DLL (or mex file, as it is called by MATLAB) are supplied, including source, with a small MATLAB GUI sample program that demonstrates the calls to the mex file. The mex file is called 'spectrometer.mexw32' c.q. 'spectrometer.mexw64', the sample program consists of 'test.m', together with 'test.fig'. The spectrometer DLL is called in turn by the mex file, and needs to be present in the directory together with the .m / .fig file and the 'spectrometer.mexwxx' file.
Note that all 64 bit versions of the AvaSpec DLL (avaspecx64.dll and as5216x64.dll) need the Visual C++ 2008 runtime to be present on your PC. If you do not have Visual C++ 2008 installed on your PC, you need to run 'vcredist_x64.exe'.
The older AS5216x64 DLL, version 2.x needs the Qt4 runtime DLL as well, which is called 'QtCore4.dll', and whicht is best located in the same directory as 'as5216x64.dll'.

The 32 bit mex file was compiled in Visual C++ 2003 and tested on MATLAB 7.1, R14 SP3. This is the latest version of this compiler that is usable with this version of MATLAB. If you have a later version of MATLAB, please use the compiler that it recommends in the mex setup, as the link libraries supplied with MATLAB may only support specific versions. E.g. MATLAB 7.1 does not supply link libraries for Visual C++ 2005 or 2008, as these versions were not yet released. Using the VC++ 2003 link libraries with VC++ 2008 results in a mex file that MATLAB 7.1 will not load.

The 64 bit mex file was compiled in Visual C++ 2008. It was tested on the 64 bit version of MATLAB R2010a.

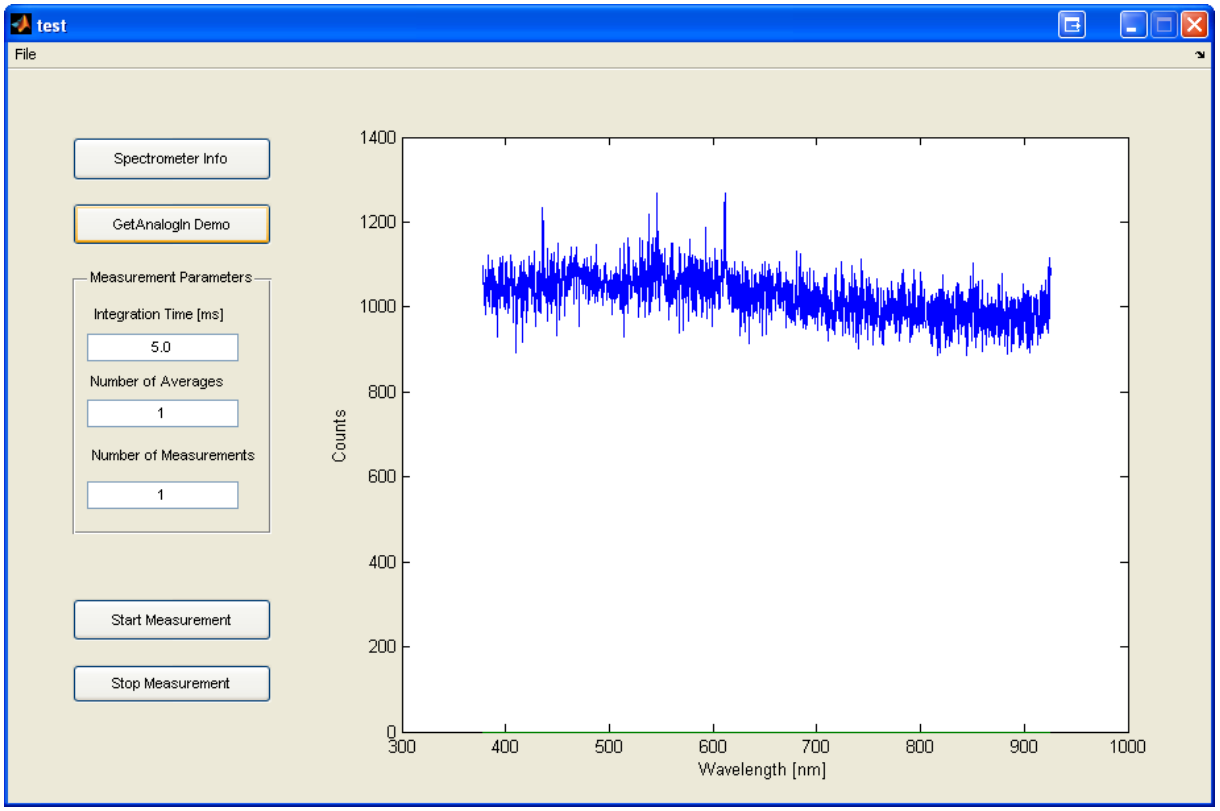
A project file for the Visual C++ IDE is supplied, which enables you to easily compile both debug and release versions of the mex file. The project file is located in the 'spectrometer' subdirectory, and called 'spectrometer.vcproj'.
The release version of the mex file shows no messages in MATLAB's command window when executing, the debug version does show these debug messages. The source code of the mex file shows how to use these messages, which can be invaluable in debugging.
You can also compile the mex file from inside the MATLAB environment. To do this, run the 'compile.m' file from the 'matlab_compile' subdirectory. You will first have to setup the mex environment in MATLAB. Please consult the MATLAB help on how to do this. Note that you will still need Visual C++ to compile the mex file, other C++ compilers are not supported.

1) Single spectrometer channel DLL

This mex file supports a single spectrometer channel only. The spectrometer handle is not exported from the DLL, which limits the amount of parameters used in the different calls, and simplifies programming. This DLL automatically opens the first spectrometer channel found, and uses the AVS_PollScan call to wait for incoming data when executing the 'measure' command.

(A version for multiple spectrometers is included in the multichannel.zip archive that is included. This version does export the spectrometer handle, which is then needed in most commands.)

The test program can be started by running the ‘test.m’ program. It will automatically open the first AvaSpec spectrometer found.



Pressing the ‘Spectrometer Info’ button will generate some information about the spectrometer that is attached:

Pressing the ‘GetAnalogIn Demo’ button will show a reading of the USB voltage that is supplied to the spectrometer (AS5216 only, the AS7010 will return 0 V):

You can enter some values in the edit boxes and start scanning by pressing the ‘Start Measurement’ button. The ‘Stop Measurement’ button can be used to stop scanning before the number of scans that is entered has been reached.

DLL calls that were implemented in the mex file:

The first five calls were implemented in the mex file internally only. They do not have to be explicitly called from your .m file:

- AVS_Init
- AVS_Done
- AVS_GetList,
- AVS_Activate
- AVS_Deactivate

The following calls can be made from your .m file, a command was implemented for each call:

- AVS_PrepareMeasure	measconfig
- AVS_Measure	measure
- AVS_GetLambda	getlambda
- AVS_GetNumPixels	getnumpixels
- AVS_GetParameter	getparameter
- AVS_PollScan	both in ‘measure’ and ‘ready’
- AVS_GetScopeData	getdata
- AVS_GetSaturatedPixels	getsaturated
- AVS_GetAnalogIn	getanalogin
- AVS_GetDigIn	getdigin
- AVS_GetVersionInfo	getversion
- AVS_SetParameter	setparameter
- AVS_SetAnalogOut	setanalogout
- AVS_SetDigOut	setdigout
- AVS_SetPwmOut	setpwmout
- AVS_SetSensitivityMode	setsensitivitymode
- AVS_StopMeasure	stop
- AVS_SetPrescanMode	setprescanmode
- AVS_UseHighResAdc	usehighres

The AVS_PollScan call is implemented in the ‘measure’ command, and does not have to be separately called. The AVS_GetParameter and AVS_SetParameter commands have a minimal implementation only. AVS_GetParameter can only read the Friendly Name and Sensor Type, AVS_SetParameter can only set the Friendly Name. The commands can easily be expanded in the C++ source provided. Please use the AVS_SetParameter command with great caution, as it is easy to overwrite essential parameters of the spectrometer.

The following measurement parameters are supported, a variable was defined for each:

- config.m_IntegrationTime	IntegrationTime
- config.m_StartPixel	StartPixel
- config.m_StopPixel	StopPixel
- config.m_IntegrationDelay	IntegrationDelay
- config.m_NrAverages	NrAverages
- darkcorrection.m_Enable	CorDynDark
- smoothing.m_SmoothPix	Smoothing
- trigger.m_Mode	TriggerMode
- trigger.m_Source	TriggerSource
- trigger.m_SourceType	TriggerSourceType
- config.m_SaturationDetection	SaturationDetection
- controlsettings.m_StoreToRam	NrStoreToRam

2) Multiple spectrometer channel DLL

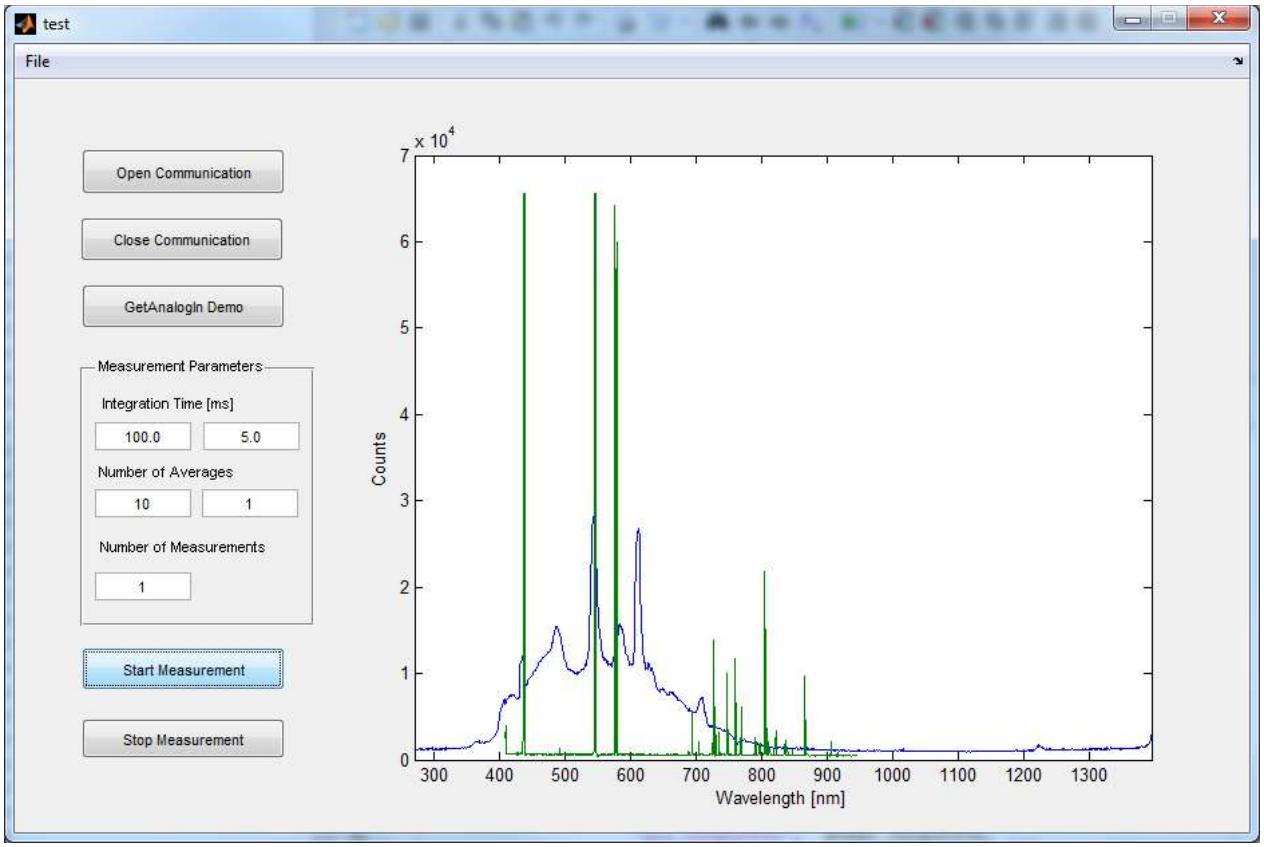
This DLL exports the spectrometer handle to MATLAB. You will need to explicitly open each channel, and add the spectrometer handle to each call. Waiting for the arrival of new data is also done inside MATLAB.

The five calls mentioned above are now implemented with a command, and need to be called from MATLAB.

- AVS_Init init
- AVS_Done done
- AVS_GetList, getlist
- AVS_Activate activate
- AVS_Deactivate deactivate

The 'measure' call now does not include waiting for a new scan with the AVS_PollScan call. You must now explicitly call 'ready' in a loop to wait for the arrival of scans after sending the 'measure' call.

A sample MATLAB program for two spectrometers is included in the multichannel subdirectory. The program will wait for the arrival of scan data from both channels, before displaying the charts. Triggering and hardware synchronization are not implemented in this demo, but should be perfectly possible.



MATLAB is a registered trademark of The MathWorks, Inc.
Visual C++ is a trademark of the Microsoft Corporation.

Avantes would like to thank Dr. Joris E. Coppens from the Netherlands Institute for Neuroscience for supplying a preliminary version of the mex file, on which our work was based. Please note that all support questions should be put to Avantes, and NOT to him.