MEDAQLib
Quick reference

Micro-Epsilon Data Acquisition Library

Content

MICRO-EPSILON

1. Introduction
2. Advantages against Low-Level Programming
3. Supported programming languages, hardware interfaces and sensors
4. System requirements
5. Internal structure of the MEDAQLib
6. Basic structure of an application
7. Example program

1. Introduction

- **Why necessary: Wide range of sensors types**
  - Many different sensors with completely different protocols
  - Diverse interface possibilities, fractional multiple per sensor
  - Complex emdedding of drivers (e.g. Windows sockets, USB)

- **What is MEDAQLib?**
  - Free Software and Driver Library for software developer
  - Applicable for all programming languages, which can use DLLs
  - Receiving, buffering and processing measured values
  - Setup and controlling of sensors via a unified programming interface

- **What is MEDAQLib not?**
  - Ready to use Measurement Software with a user interface

2. Advantages against Low-Level Programming

- Data transfer protocol and converting formulas don‘t need to be learned
- Sensor commands don‘t need to be generated manually, automatic parameter range check
- All commands of each sensor will be build equally
- Available examples for many programming languages and for many sensors

→ Fast success at implementation

→ High reuseability of code when changing interface or sensor

3. Supported Programming languages, hardware interfaces and sensors

Software	Programming language			**C** **C++**	**VB 6.0** **VBA** (e.g. Excel)	**.net** **(VB, C#)**	**Delphi**	**MatLab** **LabView** **...**	**ICONNECT**	**...**				
	Driver	**MEDAQLib** (DLL interface)												
		RS232 driver (Windows)			**IF2004_USB driver**	**WinUSB**		**IF2004 driver**	**IF2008 driver**	**TCP/IP client/server, UDP** (Windows socket)				
Hardware	Computer	**native RS232 interface**	**USB interface**					**IF2004 card**	**IF2008 card**	**Ethernet interface**				
	External adapter		**RS232 / USB conver-ter**	**RS485 / USB converter**	RS422 / USB converter (e.g. **IF2001/USB**)	**IF2004/USB**				**IF2008/ ETH**		**RS232 / Ethernet converter** (e.g. Lantronix X-Port)	**RS422 / Ethernet converter** (e.g. Lantronix EDS4100)	
	Sensor	RS232		RS485	RS422		USB	RS422	RS422		Ethernet	RS232	RS422	
		optoNCDT ILR (**ILR118x**, **ILR1191**), optoNCDT (**ILD1401**, **ILD1402**), confocalDT (**IFD2401**, **IFD2431**), optoCONTROL (**ODC1202**, **ODC1220**, **ODC2500**, **ODC2600**)		**MEBus** sensors, capaNCDT (**DT6120**)	optoNCDT ILR (**ILR110x_115x**, **ILR118x**, **ILR1191**), optoNCDT (**ILD1302**, **ILD1320**, **ILD1402**, **ILD1420**, **ILD1700**, **ILD1750**, **ILD1900**, **ILD2200**, **ILD2220**, **ILD2300**), confocalDT (**IFD2401**, **IFD2431**, **IFD2421**, **IFD2422**, **IFD2445**, **IFD2451**, **IFD2461**, **IFD2471**), optoCONTROL (**ODC2500**, **ODC2520**, **ODC2600**), colorCONTROL (**ACS7000**), **CSP2008**, **C-Box**		confocalDT (**IFD2401**, **IFD2431**), **C-Box**	optoNCDT (**ILD1302**, **ILD1320**, **ILD1402**, **ILD1420**, **ILD1700**, **ILD1750**, **ILD1900**, **ILD2200**, **ILD2220**, **ILD2300**), confocalDT (**IFD2421**, **IFD2422**, **IFD2445**, **IFD2451**, **IFD2461**, **IFD2471**) optoCONTROL (**ODC2500**, **ODC2520**, **ODC2600**), colorCONTROL (**ACS7000**), **CSP2008**, **C-Box**	same sensors as at IF2004/USB		confocalNCDT (**IFD2421**, **IFD2422**, **IFD2445**, **IFD2451**, **IFD2461**, **IFD2471**), optoNCDT (**ILD2300**), eddyNCDT (**DT306x**, **DT307x**, **DT3100**), capaNCDT (**DT6100**, **DT6200**, **DT6500**, **DT6530**), combiSENSOR (**KSS6380**, **KSS64xx**), optoCONTROL (**ODC2520**), colorCONTROL (**ACS7000**), **CSP2008**, **Eth_IF1032**, **C-Box**, **thicknessSENSOR**	same sensors as at native RS232 interface	same sensors as at IF2004/ USB	

4. System requirements

- MEDAQLib works at Windows NT, 2000, XP, Vista, Windows7, 8, 10

- RS232 and TCP/IP from Windows NT
- IF2001_USB and IF2004_USB from Windows 2000
- USB (native over WinUSB) and IF2008 from Windows XP

- RS232, IF2001_USB, IF2004_USB, USB (native over WinUSB), IF2008 and TCP/IP for 32 Bit and 64 Bit
- IF2004 for 32 Bit only

- other converters depending on driver (supplied by manufacturer)

5. Internal Structure of MEDAQLib

- Application interface
  - Equal for all sensors and interfaces

- Sensor layer
  - Knows the data protocol and converts binary data into measured values
  - Knows the communication protocol, creates sensors commands and interprets the sensor answer

- Ring buffer (between the layers)
  - Stores binary data and sensor answer coming from interface layer
  - Transfers it to the sensor layer

- Interface layer
  - Communicates with the underlying driver
  - Transmitts data from the sensor layer to the sensor, receives data from the sensor and stores it in ring buffer

6. Basic structure of an application

- Create a sensor instance (an own instance for each sensor)
- Set the interface parameters (e.g. “RS232“, “IF2008“ oder “TCP/IP“)
- Open the interface
- Query sensor settings
- If neccessary parameterize the sensor (always set command and each parameter using an own function call)

- Continous reading of sensor values (either complete data blocks or polling latest value for controlling)

- Close the interface
- Release the sensor instance

Each function returns an error value (usually ERR_NOERROR).

If there is an error it must be read.

7. Example program

Example how to open a sensor ILD1402 over RS232 and sending commands:

```
// Creating the instance
DWORD instance= CreateSensorInstance (SENSOR_ILD1402);

// Setting up interface parameter and opening the interface
ERR_CODE err= SetParameterString (instance, "IP_Interface", "RS232");
err= SetParameterString (instance, "IP_Port", "COM1");
err= OpenSensor (instance);

// Receiving sensor settings and getting sensor measurement range
err= SetParameterString (instance, "S_Command", "Get_Settings");
err= SensorCommand (instance);
double range;
err= GetParameterDouble (instance, "SA_Range", &range);

// Set the sampling rate
err= SetParameterString (instance, "S_Command", "Set_Speed");
err= SetParameterInt (instance, "SP_Speed", 1); // 1 means 1.0 kHz
err= SensorCommand (instance);
```

7. Example program

Example for getting number of available values:

```
int avail;
err= DataAvail (instance, &avail);
// avail contains now the number of available values.
```

Example for reading continous data:

```
const int blockSize= 1000;
int avail, rawData[blockSize];
double scaledData[blockSize];
err= TransferData (instance, rawData, scaledData, blockSize, &read);
// read contains now the number of available values.
// rawData contains the raw values received from sensor.
// scaledData contains scaled values.
```

Example for polling latest measurement:

```
int rawData;
double scaledData;
err= Poll (instance, &rawData, &scaledData, 1/*only one value*/);
// rawData contains the latest raw value received from sensor.
// scaledData contains the latest scaled value.
```

7. Example program

Example for closing the interface and releasing sensor instance:

```
// Close the interface
err= CloseSensor (instance);

// Release the instance
err= ReleaseSensorInstance (instance);
```

Example for reading error text:

```
char errText[1024];
GetError (instance, errText, 1024);
```