

MEDAQLib Schnellstartanleitung

Micro-Epsilon Data Acquisition Library



Content

1. Einleitung
2. Vorteile gegenüber Low-Level Programmierung
3. Unterstützte Programmiersprachen, Hardwareschnittstellen und Sensoren
4. System Voraussetzungen
5. Interner Aufbau der MEDAQLib
6. Grundsätzlicher Aufbau einer Applikation
7. Programmierbeispiel

1. Einleitung

- **Ausgangsproblem: Sensorenvielfalt**
 - Viele unterschiedliche Sensoren mit komplett unterschiedlichen Protokollen
 - Diverse Anschlussmöglichkeiten, teilweise mehrere pro Sensor
 - Komplexe Einbindung der Treiber (z.B. Windows sockets, USB)
- **Was ist MEDAQLib ?**
 - kostenlose Software- und Treiberbibliothek für Programmentwickler
 - Verwendbar für alle Programmiersprachen, die DLLs einbinden können
 - Empfang, Pufferung und Aufbereitung von Messdaten
 - Einstellung und Steuerung von Sensoren über einheitliche Programm-Schnittstelle
- **Was ist MEDAQLib nicht ?**
 - fertiges Messprogramm mit Oberfläche und Benutzerinterface

2. Vorteile gegenüber Low-Level Programmierung

- Datenübertragungsprotokoll und Konvertierungsformeln müssen nicht erlernt werden
 - Sensor Kommandos müssen nicht „per Hand“ aufgebaut werden, automatische Wertebereichsüberprüfung
 - Alle Kommandos aller Sensoren werden in gleicher Art und Weise aufgebaut
 - Vorhandene Beispiele in vielen Programmiersprachen für viele Sensoren
-
- ➔ Schneller Erfolg bei der Implementierung
 - ➔ Hohe Wiederverwendbarkeit des Programmcodes bei Wechsel der Schnittstelle oder des Sensors

3. Unterstützte Programmiersprachen, Hardware-schnittstellen und Sensoren

Software	Programmier- sprache														
	Treiber	MEDAQLib (DLL-Schnittstelle)													
	Computer	RS232-Treiber (Windows)		IF2004_USB Treiber		WinUSB		IF2004 Treiber		IF2008 Treiber		TCP/IP Client/Server, UDP (Windows Socket)			
Hardware	Externer Adapter	native RS232 Schnitt- stelle	USB Schnittstelle					IF2004 Karte		IF2008 Karte		Ethernet Schnittstelle			
	Sensor		RS232 / USB Konver- ter	RS485 / USB Konverter	RS422 / USB Konverter (z.B. IF2001/USB)	IF2004/USB				IF2008/ ETH		RS232 / Ethernet Konverter (z.B. X-Port von Lantronix)	RS422 / Ethernet Konverter (z.B. EDS4100 von Lantronix)		
		RS232	RS485	RS422		USB		RS422		RS422		Ethernet		RS232	RS422
		optoNCDT ILR (ILR118x, ILR1191), optoNCDT (ILD1401, ILD1402), confocalDT (IFD2401, IFD2431), optoCONTROL (ODC1202/20, ODC2500, ODC2600)	MEBus Sensoren, capaNCDT (DT6120)	optoNCDT ILR (ILR110x_115x, ILR118x, ILR1191, ILR2250), optoNCDT (ILD1220, ILD1302, ILD1320, ILD1402, ILD1420, ILD1700, ILD1750, ILD1900, ILD2200, ILD2220, ILD2300), confocalDT (IFD2401, IFD2431, IFD2421, IFD2422, IFD2445, IFD2451, IFD2461, IFD2471), interferoMETER (IMC5400/10, IMC5600), optoCONTROL (ODC2500, ODC2520, ODC2600), colorCONTROL (ACS7000, CFO100/200, MFA- 7/14/21/28), CSP2008, C-Box Analog			confocalDT (IFD2401, IFD2431), C-Box		optoNCDT (ILD1220, 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 Analog		alle Sensoren wie bei IF2004/USB		confocalNCDT (IFD2421, IFD2422, IFD2445, IFD2451, IFD2461, IFD2471), optoNCDT (ILD2300), eddyNCDT (DT306x/7x, DT3100), capaNCDT (DT6100, DT62xx, DT65xx), interferoMETER (IMC5400/10, IMC5600), combiSENSOR (KSS6380, KSS64xx), optoCONTROL (ODC2520), colorCONTROL (ACS7000, CFO100/200), CSP2008, IF1032/ETH, IF2008/ETH, C-Box, thicknessSENSOR		alle Sensoren wie bei nativer RS232 Schnittstelle

4. System Voraussetzungen

- MEDAQLib läuft unter Windows NT, 2000, XP, Vista, Windows 7, 8, 10
- RS232 und TCP/IP ab Windows NT
- IF2001_USB und IF2004_USB ab Windows 2000
- USB (nativ über WinUSB) und IF2008 ab Windows XP
- RS232, IF2001_USB, IF2004_USB, USB (nativ über WinUSB), IF2008 und TCP/IP für 32 Bit und 64 Bit
- IF2004 nur für 32 Bit
- andere Konverter je nach Treiber vom Hersteller

5. Interner Aufbau der MEDAQLib

- **Applikationsschnittstelle**
 - Für alle Sensoren und Interfaces identisch
- **Sensor Schicht**
 - Kennt das Datenprotokoll und konvertiert die Binärdaten in Messwerte
 - Kennt das Kommunikationsprotokoll, baut Sensorbefehle auf und interpretiert die Sensorantworten
- **Ringpuffer (zwischen den Schichten)**
 - Speichert die binären Daten sowie Antworten vom Sensor aus der Interface Schicht
 - Gibt diese weiter an die Sensor Schicht
- **Interface Schicht**
 - Kommuniziert mit dem darunter liegenden Treiber
 - Sendet Daten aus der Sensorschicht an den Sensor, empfängt Daten vom Sensor und legt diese im Ringpuffer ab

6. Grundsätzlicher Aufbau einer Applikation

- Erzeugen einer Sensorinstanz (bei mehreren Sensoren mehrere Instanzen)
- Einstellen der Schnittstellenparameter (z.B. "RS232", "IF2008" oder "TCP/IP")
- Öffnen der Schnittstelle
- Abfragen der Sensoreinstellungen
- Ggf. Parametrierung des Sensors (immer Kommando und dessen Parameter einzeln festlegen)
- Kontinuierliches Abfragen der empfangenen Messwerte (entweder komplett in Datenblöcken oder aktuellster Wert für Regelung/Steuerung)
- Schließen der Schnittstelle
- Freigeben der Sensorinstanz

Jede Funktion gibt einen Fehlerwert zurück (im Normalfall ERR_NOERROR).

Wenn ein Fehler auftritt, muss dieser ausgelesen werden

7. Programmierbeispiel

Beispiel zum Öffnen und des Sensors ILD1402 über RS232 und Senden von Kommandos:

```
// Erzeugen der Instanz
DWORD instance= CreateSensorInstance (SENSOR_ILD1402);

// Setzen der Interfaceparameter und Öffnen der Schnittstelle
ERR_CODE err= SetParameterString (instance, "IP_Interface", "RS232");
err= SetParameterString (instance, "IP_Port", "COM1");
err= OpenSensor (instance);

// Abfragen der Sensoreinstellungen und Auslesen des Wertebereichs
err= SetParameterString (instance, "S_Command", "Get_Settings");
err= SensorCommand (instance);
double range;
err= GetParameterDouble (instance, "SA_Range", &range);

// Setzen der Abtastrate
err= SetParameterString (instance, "S_Command", "Set_Speed");
err= SetParameterInt (instance, "SP_Speed", 1); // 1 entspricht 1.0 kHz
err= SensorCommand (instance);
```

7. Programmierbeispiel

Beispiel zum Abfragen der verfügbaren Messwerte:

```
int avail;  
err= DataAvail (instance, &avail);  
// In avail steht ist nun die Anzahl der verfügbaren Messwerte.
```

Beispiel zum kontinuierlichen Auslesen aller Messwerte in Blöcken:

```
const int blockSize= 1000;  
int avail, rawData[blockSize];  
double scaledData[blockSize];  
err= TransferData (instance, rawData, scaledData, blockSize, &read);  
// In read steht nun die Anzahl der gelesenen Messwerte  
// rawData enthält die Rohwerte vom Sensor  
// scaledData enthält die skalierten Messwerte.
```

Beispiel zum Auslesen der aktuellsten Messung:

```
int rawData;  
double scaledData;  
err= Poll (instance, &rawData, &scaledData, 1/*nur ein Wert*/);  
// rawData enthält den aktuellsten Rohwert vom Sensor  
// scaledData enthält den aktuellsten skalierten Messwert.
```

7. Example program

Beispiel zum Schließen der Schnittstelle und Freigeben der Sensorinstanz:

```
// Schließen der Schnittstelle  
err= CloseSensor (instance);  
  
// Freigeben der Sensorinstanz  
err= ReleaseSensorInstance (instance);
```

Beispiel zum Abfragen des Fehlertexts:

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