

SCC1-RS485/USB Sensor Cable

Driver DLL Documentation

Summary

This document describes the communication with Sensirion sensor products via the SCC1-RS485 Sensor Cable and its dedicated Microsoft Windows driver DLL.

The communication between this DLL and the RS485 sensor hardware is based on the product's Sensirion-HDLC commands. (see separate documentation)

CONTENT

This is a documentation for the SHDLC driver DLL. It contains the necessary information to use the library. Please also see the SCC1-RS485/USB Sensor Cable SHDLC Command reference for further information.

RECENT CHANGES ON THIS DOCUMENT

date	Version	Author	why
13.09.11		LWI	initial
05.01.12	3	LWI	Add Documentation Version 3.3.1 Add new status Bits 3.3.28 Add command Start Auto detection Measurement 6 Add Samplecode for C++ and C#
26.04.12	4	LWI	3.2.15, 3.2.16 Add command I2C delay 3.3.5 Add command Start continuous Measurement with Resolution 3.3.10 Add Trigger Continuous Measurement 3.3.16 Add command get last Measurement without clear 3.3.19 Add Get extended Measurement Buffer command 3.3.20 Add Get extended Buffer Size command 3.3.21 Add Clear Buffer command 3.3.30, 3.3.31 Add Advanced Measurement Configurations 3.6.2, 3.6.2 Add Autostart command
31.08.12	5	LWI	General changes in descriptions
30.10.12	6	LWI	General changes in descriptions
23.11.12	7	LWI	1 Add comment that DII is 32Bit 6.2 Add calling convention in C# Code
24.03.14	8	LWI	3.3.11 Add command TriggerMeasurementMode 3.3.26 Add command GetLastMeasurementModeDuration 3.3.32 Add command SetDetectMode 3.3.33 Add command ResetAdvancedMeasurementConfiguration
03.10.14	9	LWI	3.2.13, 3.2.14 Add command Get/SetReplyDelay
11.01.16	10	LWI	3.2.17 Add command Scan I2C Addresses 3.3.6, 3.3.8, 3.3.22 Add new commands for SF06 Sensor
16.01.17	11	LWI	3.4.2 Add command SetMeasurementType
08.05.18	12	LWI	3.3.7, 3.3.9 Add new commands for start continuous measurement 3.3.14 Add command StopContinuousMeasurementWithCommand 3.3.17 Add command GetLastMeasurementAllChannels 3.5.6 Add command GetScaleFactorAndUnit 3.2.18 Add command I2cTransceive
11.02.20	13	NSA	No updates to DLL, only to documentation.

Table Of Contents

1	INTRODUCTION	7
1.1	Required Files	7
1.2	Applicability of Commands to Different Sensor Types	7
2	SHDLCDRIVER.DLL FUNCTION REFERENCE	8
2.1	System Functions	8
2.1.1	OpenPort()	8
2.1.2	ClosePort()	8
2.2	Common Device Commands	9
2.2.1	GetDeviceInfo()	9
2.2.2	GetStringVersion()	9
2.2.3	GetVersionNbr()	10
2.2.4	DeviceReset()	10
2.2.5	GetDeviceAddress ()	10
2.2.6	SetDeviceAddress ()	11
2.2.7	FactoryReset ()	11
2.2.8	GetSystemUpTime ()	12
2.2.9	GetBaudrate ()	12
2.2.10	SetBaudrate ()	12
3	SENSORCABLEDRIVER.DLL FUNCTION REFERENCE	13
3.1	System Commands	13
3.1.1	TranslateErrorCode ()	13
3.2	Sensor Cable Commands	13
3.2.1	DeviceSelftest ()	13
3.2.2	GetTermination ()	14
3.2.3	SetTermination ()	14
3.2.4	ReadUserData ()	14
3.2.5	WriteUserData()	15
3.2.6	GetSensorVoltage ()	15
3.2.7	SetSensorVoltage ()	15
3.2.8	MeasureSensorVoltage ()	16
3.2.9	GetSensorType ()	16

3.2.10 SetSensorType ()	16
3.2.11 GetSensorAddress ()	17
3.2.12 SetSensorAddress ()	17
3.2.13 GetReplyDelay ()	17
3.2.14 SetReplyDelay ()	18
3.2.15 GetI2cDelay ()	18
3.2.16 SetI2cDelay ()	19
3.2.17 ScanI2cAddress ()	19
3.2.18 I2cTransceive ()	20
3.3 Measurement Functions	21
3.3.1 GetSensorStatus()	21
3.3.2 StartSingleMeasurement()	22
3.3.3 GetSingleMeasurementSigned/Unsigned()	22
3.3.4 StartContinuousMeasurement()	23
3.3.5 StartContinuousMeasurementWithResolution()	24
3.3.6 StartContMeasurementWithCommand()	24
3.3.7 StartContMeasurementWithCommandAdv ()	25
3.3.8 StartContMeasurementWithParameter()	25
3.3.9 StartContMeasurementWithParameterAdv()	26
3.3.10 TriggerContinuousMeasurement()	26
3.3.11 TriggerMeasurementMode ()	27
3.3.12 GetContinuousMeasurementStatus()	27
3.3.13 StopContinuousMeasurement()	28
3.3.14 StopContinuousMeasurementWithCommand ()	28
3.3.15 GetLastMeasurementSigned/Unsigned()	29
3.3.16 GetLastMeasurementWithoutClearSigned/Unsigned()	29
3.3.17 GetLastMeasurementAllChannels ()	30
3.3.18 GetMeasurementBufferSigned/Unsigned()	30
3.3.19 GetExtendedBufferSigned/Unsigned()	31
3.3.20 GetExtendedBufferSize()	31
3.3.21 ClearBuffer()	32
3.3.22 GetInterlacedBufferSigned ()	32
3.3.23 SetTotalizatorStatus ()	33
3.3.24 ResetTotalizator ()	33
3.3.25 GetTotalizatorValue ()	34

3.3.26	GetLastMeasurementModeDuration ()	34
3.3.27	GetSingleTempAndHumi()	35
3.3.28	StartAutoDetectionMeasurement()	36
3.3.29	StartStandardAutoDetectionMeasurement()	37
3.3.30	SetAdvancedMeasurementConfigurations()	38
3.3.31	GetAdvancedMeasurementConfigurations()	39
3.3.32	SetDetectMode ()	39
3.3.33	ResetAdvancedMeasurementConfiguration ()	40
3.4	Sensor settings Functions	41
3.4.1	GetMeasurementType()	41
3.4.2	SetMeasurementType()	41
3.4.3	GetResolution()	41
3.4.4	SetResolution()	42
3.4.5	GetHeaterMode()	42
3.4.6	SetHeaterMode ()	42
3.4.7	GetCalibField ()	43
3.4.8	SetCalibField ()	43
3.4.9	GetFactorySettings ()	43
3.4.10	SetFactorySettings ()	44
3.4.11	GetLinearization ()	44
3.4.12	SetLinearization()	44
3.5	Sensor Infos Functions	45
3.5.1	GetSensorPartName()	45
3.5.2	GetSensorItemNumber()	45
3.5.3	GetFlowUnit()	46
3.5.4	GetFlowUnitString()	46
3.5.5	GetScaleFactor()	46
3.5.6	GetScaleFactorAndUnit ()	47
3.5.7	GetSensorSerialNumber()	47
3.5.8	GetMeasurementDataType()	47
3.5.9	GetOffset()	48
3.6	Advanced Sensor Functions	49
3.6.1	SensorSoftReset()	49
3.6.2	GetAutoStart ()	49
3.6.3	SetAutoStart ()	50

4	DLL ERROR CODES	51
4.1	Common Device Errors	51
4.2	Device Errors	51
4.3	Common System Errors	51
4.4	System Errors	52
5	DATA TYPES	52
6	SAMPLECODE	53
6.1	C++ Sample Code	53
6.2	C# Sample Code	54

1 INTRODUCTION

This document describes the use of the 32Bit and 64Bit C-dlls to communicate with the RS485 Sensor Cable.

1.1 REQUIRED FILES

To use the SHDLC Driver with the RS485 Sensor Cable, the following files are required in the same directory as the .exe file:

- ShdlcDriver.dll
- SensorCableDriver.dll

1.2 APPLICABILITY OF COMMANDS TO DIFFERENT SENSOR TYPES

The SHDLC command reference (see separate document) lists in a table which commands apply to which sensor type. Supported sensor types include SHTxx (Humidity and Temperature), SF04 (Flow), SF05 (Flow), SF06 (Flow).

2 SHDLCDRIVER.DLL FUNCTION REFERENCE

The functions in this chapter are in the file ShdlcDriver.dll.

All functions (except TranslateErrorCode) return an error code (u32t). This error code can be translated with the function TranslateErrorCode, which returns a string with an error description.

2.1 SYSTEM FUNCTIONS

2.1.1 OPENPORT()

Description

Opens the desired port and initializes the DLL.

Prototype

```
u32t OpenPort(u8t aPortType, char* aPortConfig, u32t* aPortHandle);
```

Parameter	Meaning
aPortType	Defines which kind of port should be opened: - 0: Serial (RS232, RS485,...)
aPortConfig	String which defines the port configuration. The string format depends on the used port type: - 0 (Serial): "<ComPortName>,<Baudrate>,<EchoMode>" example: "COM1, 115200, EchoOn" EchoOn: Data sent by the master is also received by the master EchoOff: Data sent by the master is not received by the master
aPortHandle	Returned port handle

2.1.2 CLOSEPORT()

Description

Closes a port.

Prototype

```
u32t ClosePort(u32t aPortHandle);
```

Parameter	Meaning
aPortHandle	Handle of the port

2.2 COMMON DEVICE COMMANDS

2.2.1 GETDEVICEINFO()

Description

Returns information on the RS485 device as a string.

Prototype

```
u32t GetDeviceInfo(u32t aPortHandle, u8t aSlaveAdr,
                  u32t aResponseTimeoutMs, u8t aInfoType,
                  char* aInfoString, u32t aStringMaxSize) ;
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aResponseTimeoutMs	Configurable slave response timeout to enable faster searching for all devices on RS485 Bus
aInfoType	Defines the type of requested information: 1: product name 2: article code 3: serial number
aInfoString	Location where to write the information string
aStringMaxSize	Maximum number of characters allowed to write to the aInfoString location (including null-character)

2.2.2 GETVERSIONSTRING()

Description

Returns Firmware Version, Hardware Version and SHDLC Version as a string.

Prototype

```
u32t GetVersionString(u32t aPortHandle, u8t aSlaveAdr,
                     char* aVersionString, u32t aStringMaxSize)
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aVersionString	Location where to write the Version string
aStringMaxSize	Maximum number of characters allowed to write to the aVersionString location (including null-character)

2.2.3 GETVERSIONNBR()

Description

Returns Firmware Version, Hardware Version and SHDLC Version as numbers.

Prototype

```
u32t GetVersionNbr(u32t aPortHandle, u8t aSlaveAdr, u8t* aFwMajor,
                  u8t* aFwMinor, u8t* aFwDebugState, u8t* aHwMajor,
                  u8t* aHwMinor, u8t* aShdlcMajor, u8t* aShdlcMinor);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aFwMajor	Firmware version major
aFwMinor	Firmware version minor
aFwDebugState	Firmware in Debugstate if > 0
aHwMajor	Hardware version major
aHwMinor	Hardware version minor
aShdlcMajor	SHDLC version major
aShdlcMinor	SHDLC version minor

2.2.4 DEVICERESET()

Description

Perform a Reset on the Device, i.e. the Sensor Cable. This will also hard-reset any attached sensor.

Prototype

```
u32t DeviceReset(u32t aPortHandle, u8t aSlaveAdr);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address

2.2.5 GETDEVICEADDRESS ()

Description

Returns the RS485 device Address.

Prototype

```
u32t GetDeviceAddress(u32t aPortHandle, u8t aSlaveAdr, u8t* aAddress);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aAddress	Returned slave address

2.2.6 SETDEVICEADDRESS ()

Description

Set new RS485 address of the SHDLC device. If the current address of the device is unknown, a new address can be set by using this command with broadcast mode=1.

IMPORTANT NOTE: Use this command in broadcast mode only with one single device on the bus, otherwise all devices will get the same address.

Prototype

```
u32t SetDeviceAddress(u32t aPortHandle, u8t aBroadcastMode, u8t aSlaveAdr,
                     u8t aAddress);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast 0: normal (send command to aSlaveAdr and receive return) 1: Broadcast send (send with broadcast address (aSlaveAdr is ignored , no response from any slave is received. Allow sufficient time for the execution of the command before sending the next command, see the SHDLC command reference for details.) 2: Get last response (Get slave response to the previously sent broadcast command from aSlaveAdr)
aSlaveAdr	Slave address
aAddress	New address to be set

2.2.7 FACTORYRESET ()

Description

Set all device settings back to default values and do a Reset.

Prototype

```
u32t FactoryReset(u32t aPortHandle, u8t aSlaveAdr);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address

2.2.8 GETSYSTEMUPTIME ()

Description

Get the time since power on of the device in seconds.

Prototype

```
u32t GetSystemUpTime(u32t aPortHandle, u8t aSlaveAdr, u32t* aSystemUpTime);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aSystemUpTime	Returned system up time in seconds

2.2.9 GETBAUDRATE ()

Description

Get the actually set baudrate of device.

Prototype

```
u32t GetBaudrate(u32t aPortHandle, u8t aSlaveAdr, u32t* aBaudrate);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aBaudrate	Returned baudrate in baud

2.2.10 SETBAUDRATE ()

Description

Set a new baudrate.

Prototype

```
u32t SetBaudrate(u32t aPortHandle, u8t aBroadcastMode, u8t aSlaveAdr,  
u32t aBaudrate, u8t aUpdateDll);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 2.2.6)
aSlaveAdr	Slave address
aBaudrate	baudrate in baud
aUpdateDll	Define if baudrate in DLL must be updated to given value 0: Do not update Baudrate in DLL (on Master, i.e. the baudrate on the master remains unchanged. Note that in this case communication with the device is no longer possible. To restore communication, the baudrate of the master must be changed to match the baudrate of the slave) 1: Update Baudrate in DLL (on Master, i.e. overwrite the Baudrate defined in the OpenPort command with the new Baudrate specified here)

3 SENSORCABLEDRIVER.DLL FUNCTION REFERENCE

The functions in this chapter are in the file SensorCable.dll.

All functions (except TranslateErrorCode) return an error code (u32t). This error code can be translated with the function TranslateErrorCode, which returns a string with an error description.

3.1 SYSTEM COMMANDS

3.1.1 TRANSLATEERRORCODE ()

Description

This function translates an error code into an error string.

Prototype

```
const char* TranslateErrorCode(u32t aErrorCode);
```

Parameter	Meaning
aErrorCode	Code returned from a function to translate
const char*	Returned error description string if available

3.2 SENSOR CABLE COMMANDS

3.2.1 DEVICELFTEST ()

Description

Execute a selftest of the Device.

Prototype

```
u32t DeviceSelftest(u32t aPortHandle, u8t aSlaveAdr,  
u16t* aSelftestResult);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aSelftestResult	Selftest Result, bit encoded: 1: error, 0: OK Bit 0: Error with EEPROM Bit 1: Microcontroller supply voltage too high or low Bit 2: Failure on I2C Line Bit 3: Failure on sensor supply voltage

3.2.2 GETTERMINATION ()

Description

Get the current setting (enabled/disabled) of the internal termination resistor (120 Ohm).

Prototype

```
u32t GetTermination(u32t aPortHandle, u8t aSlaveAdr, u8t* aTermination);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aTermination	Returned Termination setting: 0: Termination resistor disabled 1: Termination resistor enabled

3.2.3 SETTERMINATION ()

Description

Enable / disable the internal termination resistor (120 Ohm).

Prototype

```
u32t SetTermination(u32t aPortHandle, u8t aSlaveAdr, u8t aTermination);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aTermination	Termination to be set: 0: Termination resistor disabled 1: Termination resistor enabled

3.2.4 READUSERDATA ()

Description

Read Userdata from selected Sensor Cable EEPROM block.

Prototype

```
u32t ReadUserData(u32t aPortHandle, u8t aSlaveAdr, u8t aBlockNumber,  
u8t aData[20]);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aBlockNumber	Block Number to Read [0...4]
aData	Output array of data

3.2.5 WRITEUSERDATA()

Description

Write Userdata to selected Sensor Cable EEPROM block.

Prototype

```
u32t WriteUserData(u32t aPortHandle, u8t aSlaveAdr, u8t aBlockNumber,
                  u8t aData[20]);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aBlockNumber	Block Number to write [0...4]
aData	Array of data to write

3.2.6 GETSENSORVOLTAGE ()

Description

Get the sensor supply voltage setting

Prototype

```
u32t GetSensorVoltage(u32t aPortHandle, u8t aSlaveAdr,
                     u8t* aVoltageSetting);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aVoltageSetting	Returned voltage setting, 0: 3.5V, 1: 5V

3.2.7 SETSENSORVOLTAGE ()

Description

Set the sensor supply voltage.

Prototype

```
u32t SetSensorVoltage(u32t aPortHandle, u8t aSlaveAdr,
                     u8t aVoltageSetting);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aVoltageSetting	Voltage setting, 0: 3.5V, 1: 5V

3.2.8 MEASURESENSORVOLTAGE ()

Description

Measure the output (sensor supply) voltage of the RS485 sensor cable.

Prototype

```
u32t MeasureSensorVoltage(u32t aPortHandle, u8t aSlaveAdr,
                          ft* aSensorVoltage);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aSensorVoltage	Returned sensor supply voltage in V

3.2.9 GETSENSORTYPE ()

Description

Returns the Sensor Type selected on the device.

Prototype

```
u32t GetSensorType(u32t aPortHandle, u8t aSlaveAdr, u8t* aSensorType);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aSensorType	Returned Sensortype, 0: Flow Sensor (SF04 based products) 1: Humidity Sensor (SHTxx products) 2: Flow Sensor (SF05A based products) 3: Flow Sensor (SF06 based products)

3.2.10 SETSENSORTYPE ()

Description

Select new Sensortype.

Prototype

```
u32t SetSensorType(u32t aPortHandle, u8t aSlaveAdr, u8t aSensorType);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aSensorType	New Sensortype to be set 0: Flow Sensor (SF04 based products) 1: Humidity Sensor (SHTxx products) 2: Flow Sensor (SF05A based products) 3: Flow Sensor (SF06 based products)

3.2.11 GETSENSORADDRESS ()

Description

Get the I2C Sensor Address on the cable for communication between Sensor Cable and Sensor.

Prototype

```
u32t GetSensorAddress(u32t aPortHandle, u8t aSlaveAdr,
                     u8t* aSensorAddress);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aSensorAddress	Returned I2C sensor address [0...127]

3.2.12 SETSENSORADDRESS ()

Description

Set the I2C Sensor Address on the cable for communication between Sensor Cable and Sensor.

Prototype

```
u32t SetSensorAddress(u32t aPortHandle, u8t aSlaveAdr, u8t aSensorAddress);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aSensorAddress	New I2C sensor address [0...127]

3.2.13 GETREPLYDELAY ()

Description

(for Firmware ≥1.4) Get the delay time the Sensor Cable waits before sending the reply data.

Prototype

```
u32t GetReplyDelay(u32t aPortHandle, u8t aSlaveAdr, u16t* aReplyDelay);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aReplyDelay	Returned Reply Delay in us

3.2.14 SETREPLYDELAY ()

Description

(for Firmware ≥ 1.4) Set the delay time the Sensor Cable waits before sending the reply data (in order to give the master sufficient time to switch to receiver mode).

Prototype

```
u32t SetReplyDelay(u32t aPortHandle, u8t aSlaveAdr, u16t aReplyDelay);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aReplyDelay	Reply Delay in us

3.2.15 GETI2CDELAY ()

Description

(for Firmware ≥ 1.4) Get the delay for I2C communication between Sensor Cable and Sensor.

Prototype

```
u32t GetI2cDelay(u32t aPortHandle, u8t aSlaveAdr, u16t* aI2cDelay);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aI2cDelay	Returned delay of I2C communication Value: I2C SCL Frequency 0: 600 kHz 1: 450 kHz 2 (default): 360 kHz 5: 230 kHz 10: 140 kHz 20: 80 kHz 50: 36 kHz 100: 18 kHz 200: 9 kHz 500: 3.6kHz 1000: 1.8kHz 2000: 0.9kHz

3.2.16 SetI2cDelay ()

Description

(for Firmware ≥ 1.4) Define the delay for I2C communication between Sensor Cable and Sensor.

Prototype

```
u32t SetI2cDelay(u32t aPortHandle, u8t aSlaveAdr, u16t aI2cDelay);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aI2cDelay	Delay of I2C communication, see 3.2.15

3.2.17 ScanI2cAddress ()

Description

(for Firmware ≥ 1.7) Scan all I2C addresses for Ack.

Prototype

```
u32t ScanI2cAddress(u32t aPortHandle, u8t aSlaveAdr, u8t aMode,  
u8t pAddrList[128], u8t* aNbrOfAddr);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aMode	0: Scan address Range 0...127 1: Scan address Range 8...119
pAddrList	Return Array with all acknowledged I2C Address
aNbrOfAddr	Number of found I2C Address

3.2.18 I2CTransceive ()

Description

(for Firmware ≥ 1.8) Transceives the data on I2C interface. The Transceive can contain write- and/or readheaders. If continuous measurement is active, I2C transceive is only possible, if the I2C bus isn't blocked with master hold condition. It's recommended stop continuous measurement for reliable I2C Transmissions.

Prototype

```
u32t I2cTransceive(u32t aPortHandle, u8t aSlaveAdr, u8t aI2cAddr,
                  u8t aNbrTxData, u8t aNbrRxData, u16t aTimeoutMs,
                  u8t pTxData[200], u8t pRxData[200])
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aI2cAddr	I2C Address of the targeted Device
aNbrTxData	Amount of bytes to send. This amount of bytes has to be attached to the command. Set to zero if only read header is needed.
aNbrRxData	Amount of bytes to receive. This amount of bytes have are returned as response. Set to zero if only write header is needed.
aTimeoutMs	I2C Timeout to read bytes in ms. If the read request is not acknowledged by the sensor it will be retried up to the timeout value. Same applies for clock-stretching (only for MSB bit in every read byte)
pTxData	Array with data to transmitt
pRxData	Return Array with received data

3.3 MEASUREMENT FUNCTIONS

3.3.1 GETSENSORSTATUS()

Description

Get the status of the sensor and continuous measurement. See the separate application note for a detailed description of the Auto-Detection Mode. Applies to sensor types 0, 1, 2, 3.

Prototype

```
u32t GetSensorStatus(u32t aPortHandle, u8t aSlaveAdr, u8t* aSensorStatus);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aSensorStatus	<p>Returned status of sensor or device:</p> <p>Bit 0: 0: Sensor idle 1: Sensor Busy</p> <p>Bit 1: 0: Continuous Measurement disabled, Sensor is idle or in Detect Mode 1: Continuous Measurement enabled, Sensor is in Measurement Mode</p> <p>Bit 2: (for Firmware \geq 1.3 and sensor type 0 only) 0: Auto-Detection Mode disabled 1: Auto-Detection Mode enabled</p> <p>Bit 3: (for Firmware \geq 1.3 and sensor type 0 only) 0: No confirmed measurement in Measurement Mode since last read of the Status. (Sensor has not entered Measurement Mode at all or all measurements in Measurement Mode were not confirmed.) 1: At least one measurement in Measurement Mode was confirmed and has finished since last read of the Status. This bit is set back to 0 after read out</p> <p>Bit 4: (for Firmware \geq 1.6 and sensor type 0 only) 0: No un-confirmed measurement in Measurement Mode since last read out of the Status. (Sensor has not entered Measurement Mode at all or all measurements in Measurement Mode were confirmed.) 1: At least one measurement in Measurement Mode was not confirmed and has finished since the last read of the Status. This bit is set back to 0 after read out.</p> <p>Note: Bits 3 & 4 may both be '1'. This means that both, confirmed and un-confirmed measurements in Measurement Mode have finished since last read of the Status.</p>

3.3.2 STARTSINGLEMEASUREMENT()

Description

Start a single measurement. The result must be read out with command "GetSingleMeasurement" (3.3.3) for flow sensors or "GetSingleTempAndHumi" (3.3.27) for SHTxx Sensor. Applies to sensor types 0, 1, 2.

Prototype

```
u32t StartSingleMeasurement(u32t aPortHandle, u8t aBroadcastMode,
                           u8t aSlaveAdr);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see also 2.2.6) 0: normal (send command to aSlaveAdr and receive return) 1: Broadcast send (send with broadcast address (aSlaveAdr is ignored , no response from any slave is received. Allow sufficient time for the execution of the command before sending the next command.) 2: Get last response (Get slave response to the previously sent broadcast command from aSlaveAdr)
aSlaveAdr	Slave address

3.3.3 GETSINGLEMEASUREMENTSIGNED/UNSIGNED()

Description

Read out single measurement with signed or unsigned data type. If measurement is not yet finished, error 1376 is returned. A single measurement must be started before, the completion of the measurement can be polled with this command. Applies to sensor types 0, 2.

Prototype signed

```
u32t GetSingleMeasurementSigned(u32t aPortHandle, u8t aSlaveAdr,
                                i16t* aMeasureResult);
```

Prototype unsigned

```
u32t GetSingleMeasurementUnsigned(u32t aPortHandle, u8t aSlaveAdr,
                                   u16t* aMeasureResult);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aMeasureResult	Returned measurement result as signed or unsigned integer.

3.3.4 STARTCONTINUOUSMEASUREMENT()

Description

Start continuous measurement with given interval. The measurements will be saved in a buffer. The newest 127 measurements can be read out with "Get Measurement Buffer" (3.3.18). Single measurements while continuous measurement can be read out with command "Get Last Measurement" (3.3.15). If buffering of more than 127 measurements is required, the GetExtendedBuffer (3.3.19) command has to be used. Any old measurements in buffer are cleared by the StartContinuosMeasurement command.

If the interval is set to zero, the measurement is as fast as possible. Otherwise the available minimum Interval depends on the actually set resolution in the sensor. Applies to sensor types 0, 2.

Prototype

```
u32t StartContinuousMeasurement(u32t aPortHandle, u8t aBroadcastMode,
                                u8t aSlaveAdr, u16t aInterval);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aInterval	Interval between measurements in ms 0: as fast as possible >0: Interval in ms, available minimum interval depends on the resolution. Sensor type 0: 9 Bit : min. 1ms 10 Bit : min. 2ms 11 Bit : min. 3ms 12 Bit : min. 6ms 13 Bit : min. 10ms 14 Bit : min. 20ms 15 Bit : min. 40ms 16 Bit : min. 80ms Sensor type 2: min. 1 ms for all settings.

3.3.5 STARTCONTINUOUSMEASUREMENTWITHRESOLUTION()

Description

(for Firmware ≥ 1.4) Start continuous measurement with given interval and resolution. See also StartContinuousMeasurement() 3.3.4. Applies to sensor type 0 only.

Prototype

```
u32t StartContMeasurementWithResolution(u32t aPortHandle,
                                         u8t aBroadcastMode, u8t aSlaveAdr,
                                         u16t aInterval, u8t aResolution);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aInterval	Interval between measurements in ms, 0: as fast as possible >0: Interval in ms, available minimum interval depends on the resolution. 9 Bit : min. 1ms 10 Bit : min. 2ms 11 Bit : min. 3ms 12 Bit : min. 6ms 13 Bit : min. 10ms 14 Bit : min. 20ms 15 Bit : min. 40ms 16 Bit : min. 80ms
aResolution	Measurement Resolution

3.3.6 STARTCONTMEASUREMENTWITHCOMMAND()

Description

(for Firmware ≥ 1.7) Start continuous measurement with given interval and I2C command. Applies to sensor type 3 only.

Prototype

```
u32t StartContMeasurementWithCommand(u32t aPortHandle, u8t aBroadcastMode,
                                       u8t aSlaveAdr, u16t aInterval,
                                       u16t aCommand);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aInterval	Interval between measurements in ms, 0: as fast as possible >0: Interval in ms
aCommand	16Bit I2C command to start measurement

3.3.7 STARTCONTMEASUREMENTWITHCOMMANDADV ()

Description

(for Firmware ≥1.8) Start continuous measurement with given interval, I2C command and advanced parameter configuration for continuous measurement. Applies to sensor type 3 only.

Prototype

```
u32t StartContMeasurementWithCommandAdv(u32t aPortHandle,
                                         u8t aBroadcastMode,
                                         u8t aSlaveAdr, u16t aInterval,
                                         u16t aCommand, u16t aParamterWord);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aInterval	Interval between measurements in ms, 0: as fast as possible >0: Interval in ms
aCommand	16Bit I2C command to start measurement
aParamterWord	16Bit Adv. configuration parameter for continuous measurement. See SHDLC documentation for details.

3.3.8 STARTCONTMEASUREMENTWITHPARAMETER()

Description

(for Firmware ≥1.7) Start continuous measurement with given interval, I2C command and optional I2C command parameters. Applies to sensor type 3 only.

Prototype

```
u32t StartContMeasurementWithParameter(u32t aPortHandle,u8t aBroadcastMode,
                                       u8t aSlaveAdr, u16t aInterval,
                                       u16t aCommand, u8t aParameter[],
                                       u8t aNbrOfParameter);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aInterval	Interval between measurements in ms, 0: as fast as possible >0: Interval in ms
aCommand	16Bit I2C command to start measurement
aParameter	Array with Bytes to send after I2C command
aNbrOfParameter	Number of Bytes to send after I2C command

3.3.9 STARTCONTMEASUREMENTWITHPARAMETERADV()

Description

(for Firmware ≥ 1.8) Start continuous measurement with given interval, I2C command, optional I2C command parameters and advanced parameter configuration for continuous measurement. Applies to sensor type 3 only.

Prototype

```
u32t StartContMeasurementWithParameterAdv(u32t aPortHandle,
                                           u8t aBroadcastMode,
                                           u8t aSlaveAdr, u16t aInterval,
                                           u16t aCommand, u8t aParameter[],
                                           u8t aNbrOfParameter,
                                           u16t aParamterWord);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aInterval	Interval between measurements in ms, 0: as fast as possible >0: Interval in ms
aCommand	16Bit I2C command to start measurement
aParameter	Array with Bytes to send after I2C command
aNbrOfParameter	Number of Bytes to send after I2C command
aParamterWord	16Bit Adv. configuration parameter for continuous measurement. See SHDLC documentation for details.

3.3.10 TRIGGERCONTINUOUSMEASUREMENT()

Description

(for Firmware ≥ 1.4) Start the continuous measurement with the active (advanced) measurement configuration. See also StartContinuousMeasurement(). Applies to sensor type 0 only, see SHDLC documentation for details.

Prototype

```
u32t TriggerContinuousMeasurement(u32t aPortHandle, u8t aBroadcastMode,
                                   u8t aSlaveAdr);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address

3.3.11 TRIGGERMEASUREMENTMODE ()

Description

(for Firmware ≥ 1.6) Start measurement in Measurement Mode with the current advanced measurement configuration. If the sensor is already in Measurement Mode, this command has no effect. If the sensor is currently running in Detect Mode, this command forces the sensor to enter (confirmed) Measurement Mode. Applies to sensor type 0 only.

Prototype

```
u32t TriggerMeasurementMode(u32t aPortHandle, u8t aBroadcastMode,
                           u8t aSlaveAdr);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address

3.3.12 GETCONTINUOUSMEASUREMENTSTATUS()

Description

Get the interval or/and status of the continuous measurement. Applies to sensor types 0, 2, 3.

Prototype

```
u32t GetContinuousMeasurementStatus(u32t aPortHandle, u8t aSlaveAdr,
                                    u8t* aStatus, u16t* aInterval);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aStatus	Returned status 0: continuous measurement disabled (aInterval not available) >1: continuous measurement enabled with aInterval
aInterval	Returned interval of continuous measurement, if started 0: as fast as possible >0: Interval in ms

3.3.13 STOPCONTINUOUSMEASUREMENT()

Description

Stop the continuous measurement after the current measurement is finished. Applies to sensor types 0, 2, 3.

Prototype

```
u32t StopContinuousMeasurement(u32t aPortHandle, u8t aBroadcastMode,
                               u8t aSlaveAdr);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address

3.3.14 STOPCONTINUOUSMEASUREMENTWITHCOMMAND ()

Description

(for Firmware ≥1.8) Stop continuous measurement after the current measurement with specified I2C command. Applies to sensor type 3 only.

Prototype

```
u32t StopContinuousMeasurementWithCommand(u32t aPortHandle,
                                           u8t aBroadcastMode, u8t aSlaveAdr, u16t aCommand)
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aCommand	16Bit I2C command to stop measurement.

3.3.15 GETLASTMEASUREMENTSIGNED/UNSIGNED()

Description

Read out last measurement while in continuous measurement. Use start Continuous Measurement (3.3.4) before using this command. If measurement is not started, not yet finished or no new measurement is available, error 1376 is returned. Applies to sensor types 0, 2, 3.

Prototype signed

```
u32t GetLastMeasurementSigned(u32t aPortHandle, u8t aBroadcastMode,
                              u8t aSlaveAdr, i16t* aMeasureResult);
```

Prototype unsigned

```
u32t GetLastMeasurementUnsigned(u32t aPortHandle, u8t aBroadcastMode,
                                u8t aSlaveAdr, u16t* aMeasureResult);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aMeasureResult	measurement result signed/unsigned. Note: for sensor type 3, only the first result in the sensor output (typically the flow measurement) is returned.

3.3.16 GETLASTMEASUREMENTWITHOUTCLEARSigned/UNSIGNED()

Description

(for Firmware ≥ 1.4) Read out last measurement during continuous measurement. Use start Continuous Measurement (3.3.4) before using this command. If continuous measurement is not started an error is returned. This command returns always the newest measurement, if no new value is available since the last call to this function, the same measurement is returned more than once. Before the first measurement is available, the function returns zero. Applies to sensor types 0, 2, 3.

Prototype signed

```
u32t GetLastMeasurementWithoutClearSigned(u32t aPortHandle,
                                           u8t aBroadcastMode, u8t aSlaveAdr, i16t* aMeasureResult);
```

Prototype unsigned

```
u32t GetLastMeasurementWithoutClearUnsigned(u32t aPortHandle,
                                             u8t aBroadcastMode, u8t aSlaveAdr, u16t* aMeasureResult);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aMeasureResult	Returned measurement result as signed or unsigned integer. Note: for sensor type 3, only the first result in the sensor output (typically the flow measurement) is returned.

3.3.17 GETLASTMEASUREMENTALLCHANNELS ()

Description

(for Firmware ≥ 1.8) Read out last measurement of all 3 channels during continuous measurement. Use start Continuous Measurement before using this command. If measurement is not started, not yet finished or no new measurement is available, error 1376 is returned. applies to Sensor type 3 only.

Prototype

```
u32t GetLastMeasurementAllChannels(u32t aPortHandle, u8t aBroadcastMode,
                                   u8t aSlaveAdr, u8t aClearAfterRead,
                                   i16t aMeasureResult[3])
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aClearAfterRead	0: Measurements aren't cleared after read >1: Measurements are cleared after read
aMeasureResult	Returned last measurement result of all 3 sensor measurement channels. Typically flow, temperature and aux. See sensor datasheet for details.

3.3.18 GETMEASUREMENTBUFFERSigned/UNSIGNED()

Description

Read out the newest 127 measurements and clear all measurements from the buffer. Use the "Extended Buffer commands" to work with more than 127 buffered measurements. If the returned length is 0, no new measurements are available. Applies to sensor types 0, 2.

Prototype signed

```
u32t GetMeasurementBufferSigned(u32t aPortHandle, u8t aBroadcastMode,
                                u8t aSlaveAdr, i16t aMeasureResult[127],
                                u8t* aLength);
```

Prototype unsigned

```
u32t GetMeasurementBufferUnsigned(u32t aPortHandle, u8t aBroadcastMode,
                                   u8t aSlaveAdr, u16t aMeasureResult[127],
                                   u8t* aLength);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aMeasureResult	Pointer to array to write signed or unsigned measurements
aLength	Returned number of measurements

3.3.19 GETEXTENDEDBUFFERSIGNED/UNSIGNED()

Description

Read out measurements from the extended Buffer, if more than 127 measurements are available, the oldest 127 measurement will be read out and removed from the buffer. The other measurements can be read out with the next commands. Use this command several times to read out the complete extended buffer (maximum 1000 entries). Applies to sensor types 0, 2.

Prototype signed

```
u32t GetExtendedBufferSigned(u32t aPortHandle, u8t aBroadcastMode,
                             u8t aSlaveAdr, i16t aMeasureResult[127],
                             u8t* aLength);
```

Prototype unsigned

```
u32t GetExtendedBufferUnsigned(u32t aPortHandle, u8t aBroadcastMode,
                               u8t aSlaveAdr, u16t aMeasureResult[127],
                               u8t* aLength);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aMeasureResult	Pointer to array to write signed or unsigned measurements
aLength	Returned number of measurements

3.3.20 GETEXTENDEDBUFFERSIZE()

Description

Return the actual number of measurements in the extended buffer. Applies to sensor types 0, 2, 3.

Prototype

```
u32t GetExtendedBufferSize(u32t aPortHandle, u8t aBroadcastMode, u8t
aSlaveAdr, u32t* aBufferSize);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aBufferSize	Returned number of measurements in extended Buffer

3.3.21 CLEARBUFFER()

Description

Clear all measurements from the buffer. Applies to sensor types 0, 2, 3.

Prototype

```
u32t ClearBuffer(u32t aPortHandle, u8t aBroadcastMode, u8t aSlaveAdr);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address

3.3.22 GETINTERLACEDBUFFERSIGNED ()

Description

(for Firmware ≥ 1.7) Get buffer for the oldest interlaced data, with additional buffer information. Applies to sensor type 3 only. See also the SHDLC documentation for further details.

Prototype signed

```
u32t GetInterlacedBufferSigned(u32t aPortHandle, u8t aBroadcastMode,
                               u8t aSlaveAdr,
                               i16t aInterlacedMeasurements[127],
                               u8t* aLength, u32t* aNbrOfLostData,
                               u16t* aNbrOfRemainingData,
                               u16t* aNbrOfInterlacedData);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aInterlacedMeasurements	Pointer to array to write signed interlaced measurements
aLength	Return number of measurements
aNbrOfLostData	If the time between the „GetInterlacedBufferSigned“ command calls is too large, the internal ring buffer will overrun. In this case, the oldest package of values in the buffer is cleared when a new value enters. This number is a counter which counts the missing values between the function calls (number of values which were not readout by the bus master).
aNbrOfRemainingData	The number of packages which remains in the buffer after this function call (the number of returned values is limited to 120 values because the maximum allowed data part in the SHDLC frame is 255 bytes).
aNbrOfInterlacedData	Number of interlaced data per package.

3.3.23 SETTOTALIZATORSTATUS ()

Description

Enable or disable the Totalizator. The value of the Totalizator is not changed with this command. Applies to sensor types 0, 2, 3.

Prototype

```
u32t SetTotalizatorStatus(u32t aPortHandle, u8t aBroadcastMode,
                          u8t aSlaveAdr, u8t aStatus);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aStatus	Status of the Totalizator, 0: disabled, 1: enabled

3.3.24 RESETTOTALIZATOR ()

Description

Set the Totalizator value to zero, the Totalizator status (enabled/disabled) is not changed. The Totalizator can be reset anytime. Applies to sensor types 0, 2, 3.

Prototype

```
u32t ResetTotalizator(u32t aPortHandle, u8t aBroadcastMode,
                      u8t aSlaveAdr);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address

3.3.25 GETTOTALIZATORVALUE ()

Description

Get the value of the Totalizator. This value is the sum of all unscaled measurements while in continuous measurement mode. See the separate application note for details. Applies to sensor types 0, 2, 3.

Note that for sensor type 3 only the first measurement result (typically the flow measurement) is summed up in the totalizer.

Prototype

```
u32t GetTotalizatorValue(u32t aPortHandle, u8t aBroadcastMode,
                        u8t aSlaveAdr, i64t* aTotalizatorValue);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aTotalizatorValue	Value of Totalizer

3.3.26 GETLASTMEASUREMENTMODEDURATION ()

Description

(for Firmware ≥1.6) Get the duration of last confirmed and finished measurement in Measurement Mode. (Only available in AutoDetection Mode). Applies to sensor type 0 only.

Prototype

```
u32t GetLastMeasurementModeDuration(u32t aPortHandle, u8t aBroadcastMode,
                                    u8t aSlaveAdr, u32t* aDuration);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aDuration	Duration of last measurement

3.3.27 GETSINGLETEMPANDHUMI()

Description

Read out temperature and humidity from humidity sensor (SHT7x, SHT1x or SHT2x) if finished. A single measurement (3.3.2 “StartSingleMeasurement”) must be started before, the completion of the measurement can be polled with this command. The measurement with high resolution requires a time of max. 400ms(SHT1x, SHT7x) or 110ms(SHT2x), low resolution requires 100ms(SHT1x, SHT7x) or 27ms(SHT2x). If the measurement is not yet finished, error 1376 is returned. Applies to sensor type 1 only.

Prototype

```
u32t GetSingleTempAndHumi(u32t aPortHandle, u8t aSlaveAdr,
                          ft* aTemperature, ft* aHumidity);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aTemperature	Returned temperature in °C
aHumidity	Returned humidity in % RH

3.3.28 STARTAUTODETECTIONMEASUREMENT()

Description

(for Firmware ≥ 1.3) Start auto detection measurement for liquid flow dosing applications. This function measures with low precision/power and after detection of a flow above the detection limit, switches automatically to accurate measurement mode for the given duration. During accurate measurement the bit 1 of the Sensor Status (3.3.1) is high. After the measurement duration is finished, the bit 3 in the Sensor Status is set until the Sensor Status is read out the next time. During or after the accurate measurement is running, the measurements can be read out with Get Measurement Buffer command (3.3.18). The Totalizer (if enabled, see 3.3.23) increases with the measured values only during accurate measurement. Applies to sensor type 0 only.

Prototype

```
u32t StartAutoDetectionMeasurement(u32t aPortHandle, u8t aSlaveAdr,
                                   u16t aTriggerLimit, u32t aMeasurementDurationMs,
                                   u8t aPowerMode, u16t aSearchIntervalMs,
                                   u8t aSearchResolution, u16t aMeasurementIntervalMs,
                                   u8t aMeasurementResolution, u16t aPulseConfirmationPeriodMs);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aTriggerLimit	The trigger limit must be given in 'flow ticks', i.e. flow rate (in the sensors flow unit, e.g. ul/sec) multiplied by the sensor's scale factor. A good default value is 1000, but the optimum trigger limit depends on the sensor type, liquid type and application.
aMeasurementDurationMs	The measurement should stop as soon as possible after the flow has stopped. The total measurement duration should not exceed the dosing time by more than 0.5 seconds.
aPowerMode	Currently only one power mode is supported, only use 0 for this setting.
aSearchIntervalMs	This is the interval at which the search measurement is performed to check whether the flow exceeds the trigger limit. The standard setting is 10 ms.
aSearchResolution	This is the measurement resolution with which the search measurement is performed to check whether the flow exceeds the trigger limit. The standard setting is 10 bit.
aMeasurementIntervalMs	This is the measurement interval for the continuous measurement after a flow has been detected. The standard setting is 20 ms which allows storing a dosing pulse of up to 20 seconds maximum length in the extended measurement buffer.
aMeasurementResolution	This is the measurement resolution for the continuous measurement once a flow was detected. The standard setting is 14 bit which is the maximum setting for the standard measurement interval setting.
aPulseConfirmationPeriodMs	When the confirmation period has elapsed since the last trigger event the sensor checks once if the measured flow still exceeds the trigger limit. If the flow is below the trigger limit, the sensor switches back to low power mode immediately and status bit 3 is not set. Otherwise the sensor stays in accurate measurement mode until Measurement Duration has elapsed. The standard setting for the confirmation period is 100 ms. 0: Pulse Confirmation disabled >0: Pulse Confirmation enabled with given time

3.3.29 STARTSTANDARDAUTODETECTIONMEASUREMENT()

Description

(for Firmware ≥ 1.3) Same function as 3.3.28, but the following parameters are set to default values:

Power Setting: 0

Search Interval: 10ms

Search Resolution: 10 bit

Measurement Interval: 20ms

Measurement Resolution: 14 bit

Pulse Confirmation Period: 100 ms

Applies to sensor type 0 only.

Prototype

```
u32t StartStandardAutoDetectionMeasurement(u32t aPortHandle, u8t aSlaveAdr,
                                           u16t aTriggerLimit, u32t aMeasurementDurationMs);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aTriggerLimit	Trigger Limit
aMeasurementDurationMs	Measurement Duration

3.3.30 SETADVANCEDMEASUREMENTCONFIGURATIONS()

Description

(for Firmware ≥ 1.4) Set the advanced measurement configuration to configure continuous measurement, auto detection, and advanced measurement features. See the dedicated application note for details on the parameters.

Note: The commands 'Start Continuous Measurement', 'Start Continuous Measurement and Set Resolution', 'Start Auto Detection Measurement', and 'Start standard Auto Detection Measurement' will overwrite these settings. Applies to sensor type 0 only.

Prototype

```
u32t SetAdvancedMeasurementConfigurations(u32t aPortHandle, u8t aSlaveAdr,
                                          u16t xConfiguration[19]);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
xConfiguration	Array with Configuration for Measurement xConfiguration [0] : Measurement Config 0 xConfiguration [1] : Measurement Config 1 xConfiguration [2] : Measurement Config 2 xConfiguration [3] : Measurement Config 3 xConfiguration [4] : Measurement Config 4 xConfiguration [5] : Measurement Config 5 xConfiguration [6] : Measurement Config 6 xConfiguration [7] : On Trigger Confirmation Time [ms] xConfiguration [8] : Measurement Duration High word xConfiguration [9] : Measurement Duration Low word [ms] xConfiguration [10] : Off Trigger Confirmation Time [ms] xConfiguration [11] : On Threshold [Ticks] xConfiguration [12] : Off Threshold [Ticks] xConfiguration [13] : High Range [Ticks] xConfiguration [14] : Low Range [Ticks] xConfiguration [15] : Lowest calibrated Flow [Ticks] xConfiguration [16] : Detection Period Time [ms] xConfiguration [17] : Measurement Period Time [ms] xConfiguration [18] : Measurement Selector

3.3.31 GETADVANCEDMEASUREMENTCONFIGURATIONS()

Description

(for Firmware ≥ 1.4) Get the actually set measurement configuration.

Note: the modes 'Continuous Measurement', 'Auto Detection Measurement Advanced' and 'Standard Auto Detection Measurement' are internally mapped to special cases of the advanced configuration. Their parameter settings can be read out with this command as well. Applies to sensor type 0 only.

Prototype

```
u32t GetAdvancedMeasurementConfigurations(u32t aPortHandle, u8t aSlaveAdr,
                                           u16t xConfiguration[19]);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
xConfiguration	Returned array with Configuration for Measurement, for definition see 3.3.30, SetAdvancedMeasurementConfigurations()

3.3.32 SETDETECTMODE ()

Description

(for Firmware ≥ 1.6) Enable/Disable the Detect Mode while in auto detection mode. Applies to sensor type 0 only.

Prototype

```
u32t SetDetectMode(u32t aPortHandle, u8t aBroadcastMode, u8t aSlaveAdr,
                   u8t aEnabled);
```

Parameter	Meaning
aPortHandle	Handle of the port
aBroadcastMode	Define mode for broadcast (see 3.3.2)
aSlaveAdr	Slave address
aEnabled	0: Detect Mode disabled. The auto detection is deactivated and the Sensor measures always in Measurement Mode. 1: Detect Mode enabled. The sensor will switch from Measurement Mode to Detect Mode according to the criteria defined in the Advanced Measurement Configuration.

3.3.33 RESETADVANCEDMEASUREMENTCONFIGURATION ()

Description

(for Firmware ≥ 1.6) Reset the Advanced Measurement Configuration to its default value (continuous Measurement with sensor's default settings at 100 ms sampling time) and perform a reset of the sensor. Applies to sensor type 0 only.

Prototype

```
u32t ResetAdvancedMeasurementConfiguration(u32t aPortHandle,  
                                           u8t aSlaveAdr);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address

3.4 SENSOR SETTINGS FUNCTIONS

3.4.1 GETMEASUREMENTTYPE()

Description

Get the type of measurement. Applies to sensor types 0, 2.

Prototype

```
u32t GetMeasurementType(u32t aPortHandle, u8t aSlaveAdr,
                        u8t* aMeasureType);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aMeasureType	Returned Measurement Type (0: Flow, 1: Temperature, 2: VDD)

3.4.2 SETMEASUREMENTTYPE()

Description

Set the type of measurement. Applies to sensor types 0, 2.

Prototype

```
u32t SetMeasurementType(u32t aPortHandle, u8t aSlaveAdr, u8t aMeasureType);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aMeasureType	Measurement Type (0: Flow, 1: Temperature, 2: VDD)

3.4.3 GETRESOLUTION()

Description

Get the resolution of the measurement. Applies to sensor types 0, 1, 2.

Prototype

```
u32t GetResolution(u32t aPortHandle, u8t aSlaveAdr, u8t* aResolution);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aResolution	Returned Resolution[9...16]

3.4.4 SETRESOLUTION()

Description

Set the resolution of the measurement. Applies to sensor types 0, 1, 2.

Prototype

```
u32t SetResolution(u32t aPortHandle, u8t aSlaveAdr, u8t aResolution);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aResolution	New Resolution to set [9...16]

3.4.5 GETHEATERMODE()

Description

Get the heater mode of the flow sensor. Applies to sensor type 0 only.

Prototype

```
u32t GetHeaterMode(u32t aPortHandle, u8t aSlaveAdr, u8t* aHeaterMode);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aHeaterMode	Returned heater mode: 0: always off 1: always on 2: only on for measurement

3.4.6 SETHEATERMODE ()

Description

Set the heater mode of the flow Sensor. Applies to sensor type 0 only.

Prototype

```
u32t SetHeaterMode(u32t aPortHandle, u8t aSlaveAdr, u8t aHeaterMode);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aHeaterMode	Heater mode: 0: always off 1: always on 2: only on for Measurement (Flow sensor must support this mode, else the measured flow values are incorrect)

3.4.7 GETCALIBFIELD ()

Description

Get the Calibration Field of the flow sensor. Applies to sensor type 0 only.

Prototype

```
u32t GetCalibField(u32t aPortHandle, u8t aSlaveAdr, u8t* aCalibField);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aCalibField	Returned Calibration Field setting [0...4]

3.4.8 SETCALIBFIELD ()

Description

Set the active Calibration Field of the flow sensor. Applies to sensor type 0 only.

Prototype

```
u32t SetCalibField(u32t aPortHandle, u8t aSlaveAdr, u8t aCalibField);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aCalibField	Calibration Field to set in sensor [0...4]

3.4.9 GETFACTORYSETTINGS ()

Description

Get the active Factory Settings of the flow Sensor. Applies to sensor type 0 only.

Prototype

```
u32t GetFactorySettings(u32t aPortHandle, u8t aSlaveAdr,  
                        u8t* aFactorySettings);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aFactorySettings	Returned Factory Setting [0...3]

3.4.10 SETFACTORYSETTINGS ()

Description

Set the active Factory Settings of the flow Sensor. Applies to sensor type 0 only.

Prototype

```
u32t SetFactorySettings(u32t aPortHandle, u8t aSlaveAdr,
                       u8t aFactorySettings);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aFactorySettings	Factory Setting to set in sensor [0...3]

3.4.11 GETLINEARIZATION ()

Description

Get the Linearization setting of the flow sensor. Applies to sensor types 0, 2.

Prototype

```
u32t GetLinearization(u32t aPortHandle, u8t aSlaveAdr,
                     u8t* aLinearization);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aLinearization	Returned Linearization setting [0: disabled,1: enabled]

3.4.12 SETLINEARIZATION()

Description

Enable or disable linearization of the flow measurement. Applies to sensor types 0, 2.

Prototype

```
u32t SetLinearization(u32t aPortHandle, u8t aSlaveAdr, u8t aLinearization);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aLinearization	New Linearization setting [0: disabled,1: enabled]

3.5 SENSOR INFOS FUNCTIONS

3.5.1 GETSENSORPARTNAME()

Description

Get the part name of the flow sensor. Applies to sensor types 0, 3.

Prototype

```
u32t GetSensorPartName(u32t aPortHandle, u8t aSlaveAdr,
                      char* aPartNameString, u32t aStringMaxSize);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aPartNameString	Location where to write the information string (min. length 21)
aStringMaxSize	Maximum number of characters allowed to write to the aPartNameString location (including null-character)

3.5.2 GETSENSORITEMNUMBER()

Description

Get the item number of the flow sensor. Applies to sensor types 0, 2.

Prototype

```
u32t GetSensorItemNumber(u32t aPortHandle, u8t aSlaveAdr,
                        char* aItemNumberString, u32t aStringMaxSize);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aItemNumberString	Location where to write the string (min. length 13)
aStringMaxSize	Maximum number of characters allowed to write to the aItemNumberString location (including null-character)

3.5.3 GETFLOWUNIT()

Description

Get the bit-encoded flow unit of the active calibration field of the flow sensor.
Applies to sensor types 0, 2.

Prototype

```
u32t GetFlowUnit(u32t aPortHandle, u8t aSlaveAdr, u16t* aFlowUnit);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aFlowUnit	Returned encoded flow unit

3.5.4 GETFLOWUNITSTRING()

Description

Get the flow unit of the active calibration field of the flow sensor as a string. Applies to sensor types 0, 2.

Prototype

```
u32t GetFlowUnitString(u32t aPortHandle, u8t aSlaveAdr,
                      char* aFlowUnitString, u32t aStringMaxSize);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aFlowUnitString	Location where to write the string (min. length 10)
aStringMaxSize	Maximum number of characters allowed to write to the aFlowUnitString location (including null-character)

3.5.5 GETSCALEFACTOR()

Description

Get the scale factor of the sensor for the active measurement type and calibration field. Applies to sensor types 0, 2.

Prototype

```
u32t GetScaleFactor(u32t aPortHandle, u8t aSlaveAdr, u16t* aScaleFactor);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aScaleFactor	Returned scale factor.

3.5.6 GETSCALEFACTORANDUNIT ()

Description

(for Firmware ≥ 1.8) Get the scale factor, unit and check of the sensor for the given start measurement command. Sensor type 3 only, only available on some sensor products.

Prototype

```
u32t GetScaleFactorAndUnit(u32t aPortHandle, u8t aSlaveAdr, u16t aArgument,
                          u16t* aScaleFactor, u16t* aFlowUnit,
                          u16t* aCrcResult)
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aArgument	Start command for read information
aScaleFactor	Returned scale factor
aFlowUnit	Returned flow unit
aCrcResult	Returned CRC result

3.5.7 GETSENSORSERIALNUMBER()

Description

Get the serial number of the sensor. Applies to sensor types 0, 2, 3.

Prototype

```
u32t GetSensorSerialNumber(u32t aPortHandle, u8t aSlaveAdr,
                          u32t* aSensorSerialNumber);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aSensorSerialNumber	Returned serial number

3.5.8 GETMEASUREMENTDATATYPE()

Description

Get the datatype of the flow measurements (signed or unsigned). Applies to sensor types 0, 2.

Prototype

```
u32t GetMeasurementDataType(u32t aPortHandle, u8t aSlaveAdr,
                          u8t* aDataType);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aDataType	Returned data type of measurements 0: signed (i16t) use ...Signed() Functions to read measurements 1: unsigned (u16t) use ...Unsigned() Functions to read measurements

3.5.9 GETOFFSET()

Description

Get the Offset of the current measurement type. Applies to sensor type 2 only.

Prototype

```
u32t GetOffset(u32t aPortHandle, u8t aSlaveAdr, u16t* aOffset);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aOffset	Returned offset for current measurement type

3.6 ADVANCED SENSOR FUNCTIONS

3.6.1 SENSORSOFTRESET()

Description

Execute a hard reset with the sensor and check for correct response.

Prototype

```
u32t SensorSoftReset(u32t aPortHandle, u8t aSlaveAdr);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address

3.6.2 GETAUTOSTART ()

Description

(for Firmware ≥ 1.4) Get command sequence executed after startup of device.

Prototype

```
u32t GetAutoStart(u32t aPortHandle, u8t aSlaveAdr, u8t* aNbrOfCommands,
                  u8t aData[100]);
```

Parameter	Meaning
aPortHandle	Handle of the port
aSlaveAdr	Slave address
aNbrOfCommands	Number of commands executed after Startup of Sensorcable, Autostart is disabled if aNbrOfCommands = 0.
aData	Output Array of Autostart Command Data, size must be 100

3.6.3 SETAUTOSTART ()

Description

(for Firmware ≥ 1.4) Define a command sequence to be executed upon start up of the Sensorcable. The commands must be supplied as sequence of SHDLC commands, see the respective documentation for details.

Prototype

```
u32t SetAutoStart(u32t aPortHandle, u8t aSlaveAdr, u8t aNbrOfCommands,
                  u8t aData[], u8t aSize);
```

Parameter	Meaning																
aPortHandle	Handle of the port																
aSlaveAdr	Slave address																
aNbrOfCommands	Number of commands for execute after Startup of Sensorcable, Autostart is disabled if aNbrOfCommands = 0.																
aData	Array of Data with the commands for execute Structure of Commands: <table border="1"> <thead> <tr> <th>Byte Nr</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Command ID 1</td></tr> <tr> <td>1</td><td>Nbr of Data</td></tr> <tr> <td>...</td><td>Data for command ID 1</td></tr> <tr> <td>n</td><td>Command ID 2</td></tr> <tr> <td>n+1</td><td>Nbr of Data</td></tr> <tr> <td>...</td><td>Data for command ID 2</td></tr> <tr> <td>...</td><td>...</td></tr> </tbody> </table>	Byte Nr	Description	0	Command ID 1	1	Nbr of Data	...	Data for command ID 1	n	Command ID 2	n+1	Nbr of Data	...	Data for command ID 2
Byte Nr	Description																
0	Command ID 1																
1	Nbr of Data																
...	Data for command ID 1																
n	Command ID 2																
n+1	Nbr of Data																
...	Data for command ID 2																
...	...																
aSize	Size of Command Data Array																

4 DLL ERROR CODES

4.1 COMMON DEVICE ERRORS

Error code	Error description
1	Device reported an illegal data size
2	Command not accepted from device
3	No access right on device for this command
4	Parameter out of range (report from device)

4.2 DEVICE ERRORS

Error code	Error description
32	command could not be executed because sensor is busy
33	Sensor gives no I2C acknowledge
34	CRC error while communication with sensor
35	Timeout of sensor while measurement
36	No measure is started

4.3 COMMON SYSTEM ERRORS

Error code	Error description
128	Fatal system error
129	In the Rx data stream, the start or stop byte (0x7E) is missing.
130	Too few bytes in Rx frame (frame content + checksum \geq 5 bytes).
131	The transmitted data length information in the Rx frame does not match with the number of bytes received.
132	The port configuration string has an illegal format.
133	Could not open the COM port.
134	Could not close COM port.
135	Unknown communication type of communication port.
136	Incoming checksum was wrong.
137	The device command in the received frame is not the same as sent.
138	The returned number of Data is wrong for this command
139	Illegal broadcast mode
140	One of the given arguments has an illegal size.
141	The SerialPortOverlapped class reported an error.
142	Do not use the broadcast address when calling the transceive function.
143	The maximum number of open ports which could be handled by the DLL is reached.

144	The given port handle is not valid.
145	The requested functionality is not implemented yet.
146	An error occurred while calling a windows API function.
147	A timeout occurred while waiting for the RX data.
148	The function SerialPortOverlapped.WriteData() could not write all data.
149	The COM port is not open when trying to work with it (in SerialPortOverlapped).

4.4 SYSTEM ERRORS

Error code	Error description
1024	There is no connection to ShdlcDriver.dll (library or one of it's functions could not be loaded).
1025	The returned number of Data is wrong for this command
1026	Illegal broadcast mode
1027	Wrong device command in response frame
1376	The current measure is not yet finished for read out

5 DATA TYPES

In the Documentation, an own notation for the different data types is used. Note that the DLL work with the little endian data format.

notation	C++ type	range
u8t	unsigned char	0 ... 255
i8t	signed char	-128 ... 127
u16t	unsigned int	0 ... 65535
i16t	signed int	-32768 ... 32767
u32t	unsigned long int	0 ... 4'294'967'295
i32t	signed long int	-2'147'483'648 ... 2'147'483'647
u64t	unsigned long long int	0 ... $2^{64}-1$
i64t	signed long long int	-2^{63} ... $2^{63}-1$
ft	float	6 decimals
dt	double	10 decimals
bt	bool	1/0; true/false

6 SAMPLECODE

6.1 C++ SAMPLE CODE

```
#include <windows.h>
#include <stdio.h>

/*****
 * basic types: making the size of types clear
 *****/
typedef unsigned char    u8t;    ///< range: 0 .. 255
typedef signed char      i8t;    ///< range: -128 .. +127

typedef unsigned short   u16t;   ///< range: 0 .. 65535
typedef signed short     i16t;   ///< range: -32768 .. +32767

typedef unsigned long    u32t;   ///< range: 0 .. 4'294'967'295
typedef signed long      i32t;   ///< range: -2'147'483'648 .. +2'147'483'647

typedef unsigned __int64 u64t;   ///< range: 0 .. 2^64 - 1
typedef __int64          i64t;   ///< range: -2^63 .. 2^63 - 1

typedef float            ft;     ///< range: +-1.18E-38 .. +-3.39E+38
typedef double           dt;     ///< range: .. +-1.79E+308

typedef bool             bt;     ///< values: 0, 1 (real bool used)

// Definition of commands in common Dll
typedef u32t (__cdecl *FctOpenPort) (u8t aPortType, char* aPortConfig, u32t* aPortHandle);
typedef u32t (__cdecl *FctClosePort) (u32t aPortHandle);

// Definition of commands in Sensor Cable Dll
typedef u32t (__cdecl *FctGetSensorPartName) (u32t aPortHandle, u8t aSlaveAdr, char* aPartNameString, u32t
aStringMaxSize);

int _tmain(int argc, _TCHAR* argv[])
{
    FctOpenPort OpenPort;
    FctClosePort ClosePort;
    FctGetSensorPartName GetSensorPartName;

    // Get a handle to the ShdlcDriver DLL module.
    HINSTANCE CommonLib = LoadLibrary(TEXT("ShdlcDriver.dll"));

    // If the handle is valid, try to get the function address.
    if (CommonLib != NULL)
    {
        OpenPort = (FctOpenPort)GetProcAddress(CommonLib, "OpenPort");
        ClosePort = (FctClosePort)GetProcAddress(CommonLib, "ClosePort");
    }
    else
    {
        printf("ShdlcDriver.dll not found");
        getchar();
    }

    // Get a handle to the SensorCableDriver DLL module.
    HINSTANCE SensorCableLib = LoadLibrary(TEXT("SensorCableDriver.dll"));

    // If the handle is valid, try to get the function address.
    if (SensorCableLib != NULL)
    {
        GetSensorPartName = (FctGetSensorPartName)GetProcAddress(SensorCableLib, "GetSensorPartName");
    }
    else
    {
        printf("SensorCableDriver.dll not found");
        getchar();
    }

    u32t xError;
    u32t Connection;

    // open port
    xError = OpenPort(0, "COM11, 115200, ECHOOFF", &Connection);

    // Read SensorPartName from device at port Connection and Address 0
    char Partname[256];
    xError = GetSensorPartName(Connection, 0, Partname, 256);

    printf(Partname);
}
```

```

getchar();

// close Port
xError = ClosePort(Connection);

return 0;
}

```

6.2 C# SAMPLE CODE

```

using System;
using System.Text;
using System.Runtime.InteropServices;

namespace CsSampleCode
{
    class Program
    {
        // Import of commands in common Dll
        [DllImport("ShdlcDriver.dll", EntryPoint = "OpenPort", CharSet = CharSet.Ansi,
            CallingConvention = CallingConvention.Cdecl)]
        public static extern UInt32 OpenPort(byte aPortType, string aPortConfig, out UInt32 aPortHandle);

        [DllImport("ShdlcDriver.dll", EntryPoint = "ClosePort", CharSet = CharSet.Ansi,
            CallingConvention = CallingConvention.Cdecl)]
        public static extern UInt32 ClosePort(UInt32 aPortHandle);

        // Import of commands in Sensor Cable Dll
        [DllImport("SensorCableDriver.dll", EntryPoint = "GetSensorPartName", CharSet = CharSet.Ansi,
            CallingConvention = CallingConvention.Cdecl)]
        public static extern UInt32 GetSensorPartName(UInt32 aPortHandle, byte aSlaveAdr,
            StringBuilder aSensorPartName, UInt32 aStringMaxSize);

        static void Main(string[] args)
        {
            UInt32 xPortHandle;
            UInt32 xError;

            // open Port
            xError = OpenPort(0, "COM1, 115200, ECHOOFF", out xPortHandle);

            // Read SensorPartName from device at port xPortHandle and Address 0
            StringBuilder xSensorPartName = new StringBuilder(256);
            xError = GetSensorPartName(xPortHandle, 0, xSensorPartName, 256);

            Console.WriteLine(xSensorPartName);
            Console.ReadLine();

            // close Port
            xError = ClosePort(xPortHandle);
        }
    }
}

```