SENSIRION
THE SENSOR COMPANY

SCC1-RS485/USB Sensor Cable

Driver DLL Documentation

Common DLL Commands for SF06-based Liquid Flow Sensors

Summary

This document describes the communication with Sensirion's SF06 based liquid flow sensors products via the SCC1-RS485 and SCC1-USB Sensor Cables 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)

TABLE OF CONTENTS

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 (Humidiy and Temperature), SF04 (Flow), SF05 (Flow), SF06 (Flow).

This documentation contains the most relevant commands for use with SF06-based liquid flow sensors. For a complete list of the commands available in the DLL, see the complete DLL command documentation available from your Sensirion contact.

2 SHDLCDRIVER.DLL FUNCTION REFERENCE

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

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

2.1.3 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

3 SENSORCABLEDRIVER.DLL FUNCTION REFERENCE

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

3.1 SENSOR CABLE COMMANDS

3.1.1 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

3.1.2 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

3.1.3 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	ReturnedSensortype, 0: Flow Sensor (SF04 based products) 1: Humidity Sensor (SHTxx products) 2: Flow Sensor (SF05A based products) 3: Flow Sensor (SF06 based products)

3.1.4 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

3.1.5 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]

3.1.6 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 MEASUREMENT FUNCTIONS

3.2.1 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	Returned status of sensor or device: Bit 0: 0: Sensor idle 1: Sensor Busy Bit 1: 0: Continuous Measurement disabled, Sensor is idle or in Detect Mode 1: Continuous Measurement enabled, Sensor is in Measurement Mode Bits 2..4: Relevant for sensor type 0 only.

3.2.2 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 **Error! Reference source not found.**)
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

3.2.3 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 **Error! Reference source not found.**)
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

3.2.4 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 **Error! Reference source not found.**)
aSlaveAdr	Slave address

3.2.5 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 **Error! Reference source not found.**)
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.

3.2.6 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 **Error! Reference source not found.**)
aSlaveAdr	Slave address
aBufferSize	Returned number of measurements in extended Buffer

3.2.7 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 **Error! Reference source not found.**)
aSlaveAdr	Slave address

3.2.8 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 **Error! Reference source not found.**)
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 to 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.

3.2.9 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 **Error! Reference source not found.**)
aSlaveAdr	Slave address
aStatus	Status of the Totalizator, 0: disabled, 1: enabled

3.2.10 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 **Error! Reference source not found.**)
aSlaveAdr	Slave address

3.2.11 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 **Error! Reference source not found.**)
aSlaveAdr	Slave address
aTotalizatorValue	Value of Totalizer

3.3 SENSOR INFORMATION FUNCTIONS

3.3.1 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)

3.3.2 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

3.3.3 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

3.4 ADVANCED SENSOR FUNCTIONS

3.4.1 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

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 >= 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 occured while calling a windows API function.
147	A timeout occured 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.Write(xSensorPartName);
      Console.ReadLine();

      // close Port
      xError = ClosePort(xPortHandle);

    }
  }

}
```

REVISION HISTORY

Date	Version	Change
Feb. 20	1	Initial version