

Studer Innotec Xtender Serial Communication C Library
1.5.0

Generated by Doxygen 1.8.4

Wed Jun 25 2014 13:52:56

Contents

| | | |
|----------|--|-----------|
| 1 | Studer Innotec Xtender Serial Communication C Library | 1 |
| 1.1 | Library structure | 1 |
| 1.2 | Portability | 1 |
| 2 | Data Structure Index | 3 |
| 2.1 | Data Structures | 3 |
| 3 | File Index | 5 |
| 3.1 | File List | 5 |
| 4 | Data Structure Documentation | 7 |
| 4.1 | scom_frame_flags_t Struct Reference | 7 |
| 4.1.1 | Detailed Description | 7 |
| 4.2 | scom_frame_t Struct Reference | 7 |
| 4.2.1 | Detailed Description | 8 |
| 4.3 | scom_property_t Struct Reference | 8 |
| 4.3.1 | Detailed Description | 8 |
| 4.4 | scom_service_flags_t Struct Reference | 9 |
| 4.4.1 | Detailed Description | 9 |
| 5 | File Documentation | 11 |
| 5.1 | scom_data_link.h File Reference | 11 |
| 5.1.1 | Detailed Description | 12 |
| 5.1.2 | Macro Definition Documentation | 12 |
| 5.1.2.1 | SCOM_NBR_ELEMENTS | 12 |
| 5.1.3 | Enumeration Type Documentation | 12 |
| 5.1.3.1 | scom_error_t | 12 |
| 5.1.3.2 | scom_format_t | 13 |
| 5.1.4 | Function Documentation | 13 |
| 5.1.4.1 | scom_decode_frame_data | 13 |
| 5.1.4.2 | scom_decode_frame_header | 14 |
| 5.1.4.3 | scom_encode_request_frame | 14 |
| 5.1.4.4 | scom_frame_length | 14 |

| | | |
|---------|---------------------------------|-----------|
| 5.1.4.5 | scom_initialize_frame | 14 |
| 5.2 | scom_property.h File Reference | 14 |
| 5.2.1 | Detailed Description | 15 |
| 5.2.2 | Function Documentation | 15 |
| 5.2.2.1 | scom_encode_read_property | 15 |
| 5.2.2.2 | scom_encode_write_property | 15 |
| 5.2.2.3 | scom_initialize_property | 15 |
| 5.3 | usage_examples.c File Reference | 15 |
| 5.3.1 | Detailed Description | 16 |
| 5.3.2 | Function Documentation | 16 |
| 5.3.2.1 | exchange_frame | 16 |
| 5.3.2.2 | initialize_serial_port | 16 |
| 5.3.2.3 | read_serial_port | 16 |
| 5.3.2.4 | write_serial_port | 16 |
| | Index | 17 |

Chapter 1

Studer Innotec Xtender Serial Communication C Library

This library is a reference implementation for the serial protocol for Xtender systems from Studer Innotec SA.

The protocol specification could be found in the document "Technical specification - Xtender serial protocol". The latest version of the specification could be found under "SOFTWARES AND UPDATES" at:

http://www.studer-innotec.com/?n_ulang=en&cat=download_center

1.1 Library structure

- The porting layer is defined in `scom_port_TARGET_NAME.h`, for a C99 compiler [scom_port_c99.h](#).
- [scom_data_link.h](#) implements the exchange of frames that is independent of the service.
- [scom_property.h](#) implements the `READ_PROPERTY` and `WRITE_PROPERTY` services on top of [scom_data_link.h](#)
- [usage_examples.c](#) provides a simple test implementation using a serial port viewed as a file object from `stdio` library.

1.2 Portability

The library is written in a non-blocking way that allows its use in synchronous, pooling or event-driven architectures. It is the responsibility of the user code to handle the access to the serial port and implement delays and timeouts. The library has a low memory footprint with configurable buffers and the possibility to use the same buffer for request and response. It is written in a very portable way and should be usable from small microcontrollers to a PC system.

The library targets platforms with the following requirements:

- truly ANSI C89 compiler
- big/little or mixed endianness
- 8 to 64 bit architectures
- No requirement for heap allocation functions
- no standard C library function call
- works with C++

All files apart from the `scom_port_*.h` must respect these requirements, but it doesn't mean it has to be tested for it. Patches that don't respect these requirements will usually be rejected.

The correct porting file is included at the top of [scom_data_link.h](#). Currently there is only a port for C99 compiler in [scom_port_c99.h](#) that has been tested on MS Windows x86 with GCC and Microsoft Visual Studio (compiled in C++).

Because of the interface with the serial port, execution environment, and build files are different for each target and toolchain, we do not provide any working example. However, [usage_examples.c](#) is a good base to understand how to use the library.

Studer Innotec will not respond to any questions regarding the library porting, IDE set-up, toolchain, compiler problems, etc.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

| | | |
|--------------------------------------|---|---|
| scom_frame_flags_t | Decoded content of frame_flags byte | 7 |
| scom_frame_t | Structure representing a frame | 7 |
| scom_property_t | Structure to manipulate a property with the serial protocol | 8 |
| scom_service_flags_t | Decoded content of service_flags byte | 9 |

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

| | |
|----------------------------------|----|
| scom_data_link.h | 11 |
| scom_port_c99.h | ?? |
| scom_property.h | 14 |
| usage_examples.c | 15 |

Chapter 4

Data Structure Documentation

4.1 scom_frame_flags_t Struct Reference

decoded content of frame_flags byte

```
#include <scom_data_link.h>
```

Data Fields

- int **reserved7to5**:3
- int **is_new_datalogger_file_present**:1
- int **is_sd_card_full**:1
- int **is_sd_card_present**:1
- int **was_rcc_reseted**:1
- int **is_message_pending**:1

4.1.1 Detailed Description

decoded content of frame_flags byte

The documentation for this struct was generated from the following file:

- [scom_data_link.h](#)

4.2 scom_frame_t Struct Reference

a structure representing a frame

```
#include <scom_data_link.h>
```

Data Fields

- [scom_frame_flags_t frame_flags](#)
flags specific to the datalink layer
- uint32_t [src_addr](#)
source address of this frame
- uint32_t [dst_addr](#)
destination address of this frame

- [scom_service_flags_t service_flags](#)
flags specific to the service layer
- [scom_service_t service_id](#)
identifier of the service used by this frame
- [size_t data_length](#)
length of the data payload of the frame without header and checksum
- [scom_error_t last_error](#)
last error that occurred in the frame processing
- [char * buffer](#)
buffer where the frame is build
- [size_t buffer_size](#)
maximum usable size of the buffer

4.2.1 Detailed Description

a structure representing a frame

The data buffer is variable and specified by the user with [scom_initialize_frame\(\)](#).

The documentation for this struct was generated from the following file:

- [scom_data_link.h](#)

4.3 scom_property_t Struct Reference

structure to manipulate a property with the serial protocol

```
#include <scom_property.h>
```

Data Fields

- [scom_frame_t * frame](#)
frame in which the operation are performed
- [scom_object_type_t object_type](#)
type (info, param, ...) of the object manipulated
- [uint32_t object_id](#)
identifier of the object in is type
- [uint16_t property_id](#)
identifier of the property we want to access for this particular object
- [size_t value_length](#)
length of the data (4 for INT32, ...)
- [char * value_buffer](#)
pointer with the begining of the value
- [size_t value_buffer_size](#)
maximum size that value_length can take

4.3.1 Detailed Description

structure to manipulate a property with the serial protocol

The documentation for this struct was generated from the following file:

- [scom_property.h](#)

4.4 scom_service_flags_t Struct Reference

decoded content of service_flags byte

```
#include <scom_data_link.h>
```

Data Fields

- int **reserved7to2**:6
- int **is_response**:1
- int **error**:1

4.4.1 Detailed Description

decoded content of service_flags byte

The documentation for this struct was generated from the following file:

- [scom_data_link.h](#)

Chapter 5

File Documentation

5.1 scom_data_link.h File Reference

Data Structures

- struct [scom_frame_flags_t](#)
decoded content of frame_flags byte
- struct [scom_service_flags_t](#)
decoded content of service_flags byte
- struct [scom_frame_t](#)
a structure representing a frame

Macros

- #define [SCOM_NBR_ELEMENTS](#)(array) (sizeof(array)/sizeof((array)[0]))
return the number of elements of an array (index max + 1)
- #define [SCOM_MIN](#)(a, b) ((a) < (b) ? (a) : (b))
return the minimum from two values
- #define [SCOM_MAX](#)(a, b) ((a) > (b) ? (a) : (b))
return the maximum from two values
- #define [SCOM_FRAME_HEADER_SIZE](#) 14
the size of the frame header

Enumerations

- enum [scom_error_t](#) {
[SCOM_ERROR_NO_ERROR](#) = 0x0000, [SCOM_ERROR_INVALID_FRAME](#) = 0x0001, [SCOM_ERROR_DEVICE_NOT_FOUND](#) = 0x0002, [SCOM_ERROR_RESPONSE_TIMEOUT](#) = 0x0003,
[SCOM_ERROR_SERVICE_NOT_SUPPORTED](#) = 0x0011, [SCOM_ERROR_INVALID_SERVICE_ARGUMENT](#) = 0x0012, [SCOM_ERROR_GATEWAY_BUSY](#) = 0x0013, [SCOM_ERROR_TYPE_NOT_SUPPORTED](#) = 0x0021,
[SCOM_ERROR_OBJECT_ID_NOT_FOUND](#) = 0x0022, [SCOM_ERROR_PROPERTY_NOT_SUPPORTED](#) = 0x0023, [SCOM_ERROR_INVALID_DATA_LENGTH](#) = 0x0024, [SCOM_ERROR_PROPERTY_IS_READ_ONLY](#) = 0x0025,
[SCOM_ERROR_INVALID_DATA](#) = 0x0026, [SCOM_ERROR_DATA_TOO_SMALL](#) = 0x0027, [SCOM_ERROR_DATA_TOO_BIG](#) = 0x0028, [SCOM_ERROR_WRITE_PROPERTY_FAILED](#) = 0x0029,
[SCOM_ERROR_READ_PROPERTY_FAILED](#) = 0x002A, [SCOM_ERROR_ACCESS_DENIED](#) = 0x002B,

```
SCOM_ERROR_OBJECT_NOT_SUPPORTED = 0x002C, SCOM_ERROR_MULTICAST_READ_NOT_SUPPORTED = 0x002D,
SCOM_ERROR_INVALID_SHELL_ARG = 0x0081, SCOM_ERROR_STACK_PORT_NOT_FOUND = 0x0082,
SCOM_ERROR_STACK_PORT_INIT_FAILED = 0x0083, SCOM_ERROR_STACK_PORT_WRITE_FAILED = 0x0084,
SCOM_ERROR_STACK_PORT_READ_FAILED = 0x0085, SCOM_ERROR_STACK_BUFFER_TOO_SMALL = 0x0086,
SCOM_ERROR_STACK_PROPERTY_HEADER_DOESNT_MATCH = 0x0087 }
```

scom error types

- enum `scom_service_t` { `SCOM_READ_PROPERTY_SERVICE` = 0x1, `SCOM_WRITE_PROPERTY_SERVICE` = 0x2 }

service identifier of service_id

- enum `scom_format_t` {
`SCOM_FORMAT_INVALID_FORMAT` = 0, `SCOM_FORMAT_BOOL` = 1, `SCOM_FORMAT_FORMAT` = 2,
`SCOM_FORMAT_ENUM` = 3,
`SCOM_FORMAT_ERROR` = 4, `SCOM_FORMAT_INT32` = 5, `SCOM_FORMAT_FLOAT` = 6, `SCOM_FORMAT_STRING` = 7,
`SCOM_FORMAT_DYNAMIC` = 8, `SCOM_FORMAT_BYTE_STREAM` = 9 }

data format

Functions

- void `scom_initialize_frame` (`scom_frame_t` *frame, char *buffer, size_t buffer_size)
initialize a frame structure
- void `scom_encode_request_frame` (`scom_frame_t` *frame)
encode a frame in its buffer
- void `scom_decode_frame_header` (`scom_frame_t` *frame)
decode the frame header from its buffer
- void `scom_decode_frame_data` (`scom_frame_t` *frame)
decode the frame data from its buffer
- size_t `scom_frame_length` (`scom_frame_t` *frame)
return the total frame length

5.1.1 Detailed Description

interface to send and receive scom frames (the Data Link Layer)

5.1.2 Macro Definition Documentation

5.1.2.1 `#define SCOM_NBR_ELEMENTS(array) (sizeof(array)/sizeof((array)[0]))`

return the number of elements of an array (index max + 1)

Parameters

| | |
|--------------------|------------------------|
| <code>array</code> | variable of array type |
|--------------------|------------------------|

5.1.3 Enumeration Type Documentation

5.1.3.1 enum `scom_error_t`

scom error types

Enumerator

SCOM_ERROR_NO_ERROR a value to indicate not error occurred

SCOM_ERROR_INVALID_FRAME malformed frame on the datalink layer

SCOM_ERROR_DEVICE_NOT_FOUND wrong dst_addr field

SCOM_ERROR_RESPONSE_TIMEOUT no response of the server

SCOM_ERROR_SERVICE_NOT_SUPPORTED wrong service_id field

SCOM_ERROR_INVALID_SERVICE_ARGUMENT wrong service_data

SCOM_ERROR_GATEWAY_BUSY gateway (for example XCOM-232i) busy

SCOM_ERROR_TYPE_NOT_SUPPORTED the object_type requested doesn't exist

SCOM_ERROR_OBJECT_ID_NOT_FOUND not object with this object_id was found

SCOM_ERROR_PROPERTY_NOT_SUPPORTED the property identified by property_id doesn't exist

SCOM_ERROR_INVALID_DATA_LENGTH the field property_data has an invalid number of bytes

SCOM_ERROR_PROPERTY_IS_READ_ONLY a write to this property is not allowed

SCOM_ERROR_INVALID_DATA this value is impossible for this property

SCOM_ERROR_DATA_TOO_SMALL the value is below the minimum limit

SCOM_ERROR_DATA_TOO_BIG the value is above the maximum limit

SCOM_ERROR_WRITE_PROPERTY_FAILED write is possible, but failed

SCOM_ERROR_READ_PROPERTY_FAILED read is possible, but failed

SCOM_ERROR_ACCESS_DENIED insufficient user access

SCOM_ERROR_OBJECT_NOT_SUPPORTED this object id, through existent, is not supported by the current implementation of the gateway

SCOM_ERROR_MULTICAST_READ_NOT_SUPPORTED Read operation is not supported when used on multicast addresses

SCOM_ERROR_INVALID_SHELL_ARG the command line tool used received the wrong arguments

SCOM_ERROR_STACK_PORT_NOT_FOUND the port configured to be used doesn't exist or it is not possible to open it

SCOM_ERROR_STACK_PORT_INIT_FAILED the initialization of the port failed

SCOM_ERROR_STACK_PORT_WRITE_FAILED a write operation on the port failed

SCOM_ERROR_STACK_PORT_READ_FAILED a read operation on the port failed

SCOM_ERROR_STACK_BUFFER_TOO_SMALL the buffer provided to the client stack are too small to handle the operation

SCOM_ERROR_STACK_PROPERTY_HEADER_DOESNT_MATCH the header of a property access response is not equal the response

5.1.3.2 enum scom_format_t

data format

See Also

Xtender serial protocol technical specification

5.1.4 Function Documentation

5.1.4.1 void scom_decode_frame_data (scom_frame_t * frame)

decode the frame data from its buffer

This function call be called after the reception of frame->data_length byte in frame->buffer. frame->last_error will contain SCOM_ERROR_INVALID_FRAME if the data checksum is invalid or the frame is misformed.

5.1.4.2 void scom_decode_frame_header (scom_frame_t * frame)

decode the frame header from its buffer

This function call be called after the reception of SCOM_FRAME_HEADER_SIZE byte in frame->buffer. frame->last_error will contain SCOM_ERROR_INVALID_FRAME if the checksum is invalid or the header is misformed.

5.1.4.3 void scom_encode_request_frame (scom_frame_t * frame)

encode a frame in its buffer

The frame must have been initialized with [scom_initialize_frame\(\)](#). The frame fields src_addr, dst_addr, service_id and data_length must have a valid value.

5.1.4.4 size_t scom_frame_length (scom_frame_t * frame)

return the total frame length

This function can be called after [scom_decode_frame_header\(\)](#) to know how many bytes we expect to receive.

5.1.4.5 void scom_initialize_frame (scom_frame_t * frame, char * buffer, size_t buffer_size)

initialize a frame structure

Parameters

| | |
|--------------------|--|
| <i>frame</i> | the structure to initialize |
| <i>buffer</i> | the buffer used to encode the data |
| <i>buffer_size</i> | the size of a buffer, allowing user defined size |

5.2 scom_property.h File Reference

```
#include "scom_data_link.h"
```

Data Structures

- struct [scom_property_t](#)
structure to manipulate a property with the serial protocol

Enumerations

- enum [scom_object_type_t](#) { **SCOM_USER_INFO_OBJECT_TYPE** = 0x1, **SCOM_PARAMETER_OBJECT_TYPE** = 0x2 }
different values that object_type in [scom_property_t](#) can take

Functions

- void [scom_initialize_property](#) ([scom_property_t](#) *property, [scom_frame_t](#) *frame)
initialize a [scom_property_t](#) before use
- void [scom_encode_read_property](#) ([scom_property_t](#) *property)
encode a property read request before sending it

- void [scom_encode_write_property](#) ([scom_property_t](#) *property)
encode a property write request before sending it
- void [scom_decode_read_property](#) ([scom_property_t](#) *property)
decode a property read response after reception
- void [scom_decode_write_property](#) ([scom_property_t](#) *property)
decode a property write response after reception

5.2.1 Detailed Description

interface to access to the object property of objects

5.2.2 Function Documentation

5.2.2.1 void [scom_encode_read_property](#) ([scom_property_t](#) * *property*)

encode a property read request before sending it

The fields `src_addr`, `dst_addr` must be defined in `property->frame`. `object_type`, `object_id` and `property_id` should be defined in `property`.

5.2.2.2 void [scom_encode_write_property](#) ([scom_property_t](#) * *property*)

encode a property write request before sending it

The fields `src_addr`, `dst_addr` must be defined in `property->frame`. `object_type`, `object_id` and `property_id`, `value_` - `length` and `value_buffer` should be defined in `property`.

5.2.2.3 void [scom_initialize_property](#) ([scom_property_t](#) * *property*, [scom_frame_t](#) * *frame*)

initialize a [scom_property_t](#) before use

Parameters

| | |
|-----------------|---|
| <i>property</i> | the structure to initialize |
| <i>frame</i> | an initialized scom_frame_t this will be used by the property |

5.3 usage_examples.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <scom_property.h>
```

Functions

- int [initialize_serial_port](#) ()
configure the serial port
- void [clear_serial_port](#) ()
empty the rx and tx buffer before a new exchange
- size_t [read_serial_port](#) (char *buffer, size_t byte_count)
read in a blocking way on the serial port
- size_t [write_serial_port](#) (const char *buffer, size_t byte_count)

- write in a blocking way on the serial port*
- void `close_serial_port ()`
close the serial port even if `initialize_serial_port()` failed
- `scom_error_t exchange_frame (scom_frame_t *frame)`
example code to exchange a frame and print possible error on standard output
- void `read_xt1_uBat ()`
example code to read the battery voltage on Xtender 1 and print it to the standard output
- int `main ()`

5.3.1 Detailed Description

examples of usage of the scom library

5.3.2 Function Documentation

5.3.2.1 `scom_error_t exchange_frame (scom_frame_t * frame)`

example code to exchange a frame and print possible error on standard output

Parameters

| | |
|--------------|---|
| <i>frame</i> | an initialized frame configured for a service |
|--------------|---|

Returns

a possible error that occurred or SCOM_NO_ERROR

5.3.2.2 `int initialize_serial_port ()`

configure the serial port

Returns

zero in case of an error

5.3.2.3 `size_t read_serial_port (char * buffer, size_t byte_count)`

read in a blocking way on the serial port

This function must implement the proper timeout mechanism.

Returns

- number of byte read

5.3.2.4 `size_t write_serial_port (const char * buffer, size_t byte_count)`

write in a blocking way on the serial port

This function must implement the proper timeout mechanism.

Returns

number of byte written

Index

E

exchange_frame
usage_examples.c, [16](#)

I

initialize_serial_port
usage_examples.c, [16](#)

R

read_serial_port
usage_examples.c, [16](#)

S

SCOM_ERROR_ACCESS_DENIED
scom_data_link.h, [13](#)
SCOM_ERROR_DATA_TOO_BIG
scom_data_link.h, [13](#)
SCOM_ERROR_DATA_TOO_SMALL
scom_data_link.h, [13](#)
SCOM_ERROR_DEVICE_NOT_FOUND
scom_data_link.h, [13](#)
SCOM_ERROR_GATEWAY_BUSY
scom_data_link.h, [13](#)
SCOM_ERROR_INVALID_DATA
scom_data_link.h, [13](#)
SCOM_ERROR_INVALID_DATA_LENGTH
scom_data_link.h, [13](#)
SCOM_ERROR_INVALID_FRAME
scom_data_link.h, [13](#)
SCOM_ERROR_INVALID_SERVICE_ARGUMENT
scom_data_link.h, [13](#)
SCOM_ERROR_INVALID_SHELL_ARG
scom_data_link.h, [13](#)
SCOM_ERROR_MULTICAST_READ_NOT_SUPPORTED
scom_data_link.h, [13](#)
SCOM_ERROR_NO_ERROR
scom_data_link.h, [13](#)
SCOM_ERROR_OBJECT_ID_NOT_FOUND
scom_data_link.h, [13](#)
SCOM_ERROR_OBJECT_NOT_SUPPORTED
scom_data_link.h, [13](#)
SCOM_ERROR_PROPERTY_IS_READ_ONLY
scom_data_link.h, [13](#)
SCOM_ERROR_PROPERTY_NOT_SUPPORTED
scom_data_link.h, [13](#)
SCOM_ERROR_READ_PROPERTY_FAILED
scom_data_link.h, [13](#)
SCOM_ERROR_RESPONSE_TIMEOUT
scom_data_link.h, [13](#)

SCOM_ERROR_SERVICE_NOT_SUPPORTED
scom_data_link.h, [13](#)
SCOM_ERROR_STACK_BUFFER_TOO_SMALL
scom_data_link.h, [13](#)
SCOM_ERROR_STACK_PORT_INIT_FAILED
scom_data_link.h, [13](#)
SCOM_ERROR_STACK_PORT_NOT_FOUND
scom_data_link.h, [13](#)
SCOM_ERROR_STACK_PORT_READ_FAILED
scom_data_link.h, [13](#)
SCOM_ERROR_STACK_PORT_WRITE_FAILED
scom_data_link.h, [13](#)
SCOM_ERROR_STACK_PROPERTY_HEADER_DOESNT_MATCH
scom_data_link.h, [13](#)
SCOM_ERROR_TYPE_NOT_SUPPORTED
scom_data_link.h, [13](#)
SCOM_ERROR_WRITE_PROPERTY_FAILED
scom_data_link.h, [13](#)
SCOM_NBR_ELEMENTS
scom_data_link.h, [12](#)
scom_data_link.h
SCOM_ERROR_ACCESS_DENIED, [13](#)
SCOM_ERROR_DATA_TOO_BIG, [13](#)
SCOM_ERROR_DATA_TOO_SMALL, [13](#)
SCOM_ERROR_DEVICE_NOT_FOUND, [13](#)
SCOM_ERROR_GATEWAY_BUSY, [13](#)
SCOM_ERROR_INVALID_DATA, [13](#)
SCOM_ERROR_INVALID_DATA_LENGTH, [13](#)
SCOM_ERROR_INVALID_FRAME, [13](#)
SCOM_ERROR_INVALID_SERVICE_ARGUMENT, [13](#)
SCOM_ERROR_INVALID_SHELL_ARG, [13](#)
SCOM_ERROR_MULTICAST_READ_NOT_SUPPORTED, [13](#)
SCOM_ERROR_NO_ERROR, [13](#)
SCOM_ERROR_OBJECT_ID_NOT_FOUND, [13](#)
SCOM_ERROR_OBJECT_NOT_SUPPORTED, [13](#)
SCOM_ERROR_PROPERTY_IS_READ_ONLY, [13](#)
SCOM_ERROR_PROPERTY_NOT_SUPPORTED, [13](#)
SCOM_ERROR_READ_PROPERTY_FAILED, [13](#)
SCOM_ERROR_RESPONSE_TIMEOUT, [13](#)
SCOM_ERROR_SERVICE_NOT_SUPPORTED, [13](#)
SCOM_ERROR_STACK_BUFFER_TOO_SMALL, [13](#)

- SCOM_ERROR_STACK_PORT_INIT_FAILED, [13](#)
- SCOM_ERROR_STACK_PORT_NOT_FOUND, [13](#)
- SCOM_ERROR_STACK_PORT_READ_FAILED, [13](#)
- SCOM_ERROR_STACK_PORT_WRITE_FAILED, [13](#)
- SCOM_ERROR_STACK_PROPERTY_HEADER_DOESNT_MATCH, [13](#)
- SCOM_ERROR_TYPE_NOT_SUPPORTED, [13](#)
- SCOM_ERROR_WRITE_PROPERTY_FAILED, [13](#)
- scom_data_link.h, [11](#)
 - scom_decode_frame_data, [13](#)
 - scom_decode_frame_header, [13](#)
 - scom_encode_request_frame, [14](#)
 - scom_error_t, [12](#)
 - scom_format_t, [13](#)
 - scom_frame_length, [14](#)
 - scom_initialize_frame, [14](#)
- scom_decode_frame_data
 - scom_data_link.h, [13](#)
- scom_decode_frame_header
 - scom_data_link.h, [13](#)
- scom_encode_read_property
 - scom_property.h, [15](#)
- scom_encode_request_frame
 - scom_data_link.h, [14](#)
- scom_encode_write_property
 - scom_property.h, [15](#)
- scom_error_t
 - scom_data_link.h, [12](#)
- scom_format_t
 - scom_data_link.h, [13](#)
- scom_frame_flags_t, [7](#)
- scom_frame_length
 - scom_data_link.h, [14](#)
- scom_frame_t, [7](#)
- scom_initialize_frame
 - scom_data_link.h, [14](#)
- scom_initialize_property
 - scom_property.h, [15](#)
- scom_property.h, [14](#)
 - scom_encode_read_property, [15](#)
 - scom_encode_write_property, [15](#)
 - scom_initialize_property, [15](#)
- scom_property_t, [8](#)
- scom_service_flags_t, [9](#)

U

- usage_examples.c, [15](#)
 - exchange_frame, [16](#)
 - initialize_serial_port, [16](#)
 - read_serial_port, [16](#)
 - write_serial_port, [16](#)

W

- write_serial_port
 - usage_examples.c, [16](#)