Serialized Bluetooth Low Energy

0.1

Generated by Doxygen 1.7.6.1

Thu Aug 30 2012 14:24:44

Contents

Chapter 1

SBLE - Serialized, easy-to-use Bluetooth Low Energy (BLE)

(c) 2012 Philipp Kindt <kindt@rcs.ei.tum.de>

1.1 Introduction

Bluetooth low energy is a lowlevel, packet-based radio procol. Events come in a very asynchronous way. This toolbox synchronizes the communication between BLE112 / BLED112 modules and your application and makes bluetooth low energy easy to use.

1.2 Further References

- **Introduction to Blutooth Low Energy standard:** http://e2e.ti.-com/support/low_power_rf/m/videos__files/653593/download.-aspx
- **Institute for Realtime-Computer-Systems (RCS), TUM:** http://rcs.ei.-tum.de
- **Bluegiga:**http://www.bluegiga.com
- **BLE112/BLED112/DKBLE112:**http://www.bluegiga.com/bluetooth-low-energy

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Data Structure Documentation

4.1 _sble_array Struct Reference

represents an array. Represents an array containing a pointer to allocated data and its size in bytes.

`#include <sble_array.h>`

Data Fields

- uint8_t * data
  Pointer to allocated data; "Payload" of the array.
- sble_unsigned_integer len
  The lengt of the data array in bytes.

4.1.1 Detailed Description

represents an array. Represents an array containing a pointer to allocated data and its size in bytes.

Examples:

sble_example_array.c, sble_example_client.c, sble_example_gattserver.c, and sble_example_minimal_client.c.

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

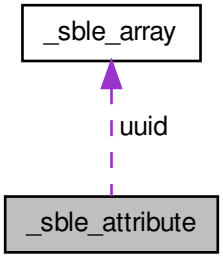
- /home/kindt/workspace/HE2mT/projects/sble/include/sble_array.h

4.2 _sble_attribute Struct Reference

Represents a GATT/ATT attribute.

`#include <sble_attribute.h>`

Collaboration diagram for _sble_attribute:



Data Fields

- sble_array * uuid

  Unique UUID of the attribute determining also its type. The UUID is the same for all nodes (including nodes by different vendors) offering the same attribute.
- uint8_t handle

  Handle on the GATT server. This handle might differ on different nodes offering the same attribute.

4.2.1 Detailed Description

Represents a GATT/ATT attribute.

Examples:

sble_example_check_for_attribute.c, sble_example_client.c, and sble_example_minimal_client.c.

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

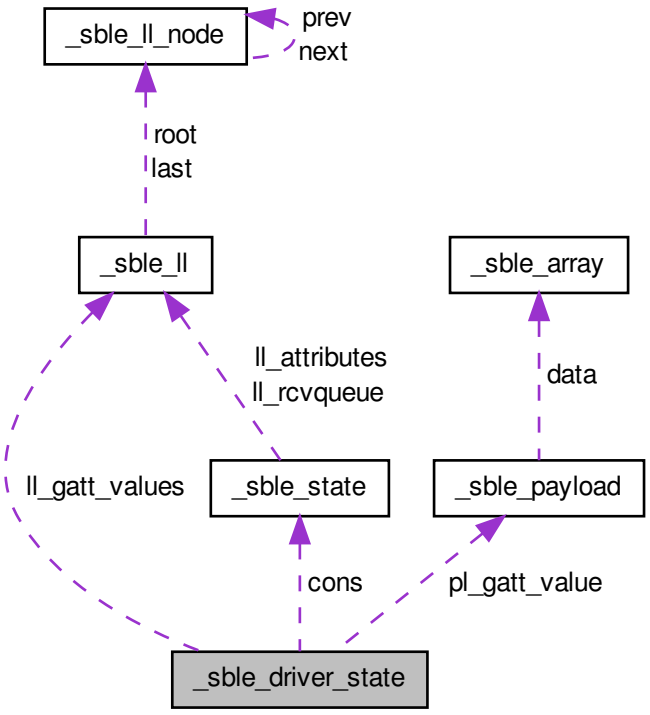
- /home/kindt/workspace/HE2mT/projects/sble/include/sble_attribute.h

4.3 _sble_driver_state Struct Reference

The global state (dstate-Variable) of SBLE.

`#include <sble_state.h>`

Collaboration diagram for _sble_driver_state:



Data Fields

- sble_state ** cons
- sble_unsigned_integer cons_length

  The lenght of the cons-array.
- uint32_t flags

  Flags. All SBLE_STATE_* - macros defined in this file.
- uint8_t flags_sched_internal

  Flags used by the scheduler internally. Do not touch!
- uint8_t addr [6]

  placeholder for an address that is filled by some command responses/events
- uint8_t addr_type

  placeholder for an address type is set by some command responses/events
- uint32_t evt_clear_list

- uint8_t current_con

  The connection the last response that occured was related to. Set by most responses to commands.
- uint32_t cons_activity_map
- sble_ll * ll_gatt_values

  reception queue for gatt values. If a remote node writes an attribute it per ATT-protocol, these values will be stored in this linked-list
- sble_payload * pl_gatt_value

  Filled by ble_rsp_attributes_read in commands.h. It is used by all calls that readout the local GATT database.
- sble_signed_integer filedescriptor

  the filedescriptor for I/O with BLE112/BLED112
- xSemaphoreHandle **sleepMain**
- xSemaphoreHandle **sleepDispatcher**
- xTaskHandle **sble_thread**

4.3.1 Detailed Description

The global state (dstate-Variable) of SBLE.

4.3.2 Field Documentation

4.3.2.1 sble_state** cons

Active connection array. Use a connection number to access connection specific data via this array.

example dstate.cons[0].addr

Examples:

sble_example_client.c.

4.3.2.2 uint32_t cons_activity_map

Bitfield for active connection. If a connection is established, the bit at the corresponding position is set Example: 01000101 => connections 0,2 and 6 are active, the others are not.

4.3.2.3 uint32_t evt_clear_list

the event autoclear list. Events on this do not have to be acknowledged by the main thread as they are acknowledged automatically. See

See also

(sble_scheduler.h) how the eal is used.

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

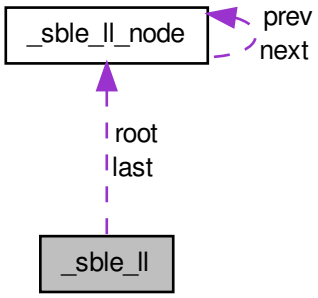
- /home/kindt/workspace/HE2mT/projects/sble/include/sble_state.h

4.4 _sble_ll Struct Reference

The structure representing a linked list.

`#include <sble_ll.h>`

Collaboration diagram for _sble_ll:



Data Fields

- struct _sble_ll_node * root

  pointer to the root node
- struct _sble_ll_node * last

  pointer to the last element in list
- sble_unsigned_integer nelements

  The number of elements in list.

4.4.1 Detailed Description

The structure representing a linked list.

Examples:

sble_example_ll.c.

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

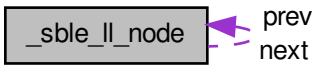
- /home/kindt/workspace/HE2mT/projects/sble/include/sble_ll.h

4.5 _sble_ll_node Struct Reference

A node of the linked list.

`#include <sble_ll.h>`

Collaboration diagram for _sble_ll_node:



Data Fields

- void * data

  Pointer to the payload, that can be of any type.
- struct _sble_ll_node * prev

  The previous node or NULL for list-root.
- struct _sble_ll_node * next

  The next node or NULL for list-top.

4.5.1 Detailed Description

A node of the linked list.

Examples:

sble_example_client.c.

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

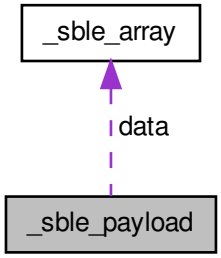
- /home/kindt/workspace/HE2mT/projects/sble/include/sble_ll.h

4.6 _sble_payload Struct Reference

Structure representing payload that is transmitted via ATT protocol.

`#include <sble_payload.h>`

Collaboration diagram for _sble_payload:



Data Fields

- uint16_t atthandle

  The attribute handle corresponding with the payload that is to be / has been transmitted.
- sble_array * data

  The actual transmission data.
- uint8_t connection

  Connection number. Each connected device will get its own number.

4.6.1 Detailed Description

Structure representing payload that is transmitted via ATT protocol.

Examples:

sble_example_client.c, sble_example_gattserver.c, and sble_example_minimal_gattserver.c.

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

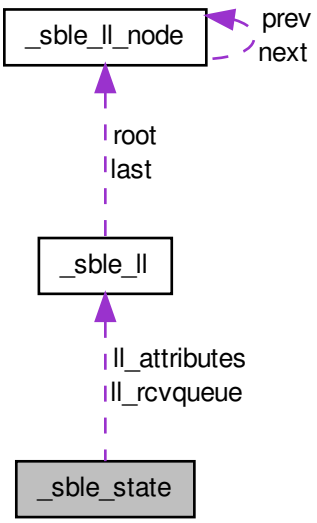
- /home/kindt/workspace/HE2mT/projects/sble/include/sble_payload.h

4.7 _sble_state Struct Reference

The sble_state struct is a per-conection-state and constitutes each entry in the cons[]-array.

`#include <sble_state.h>`

Collaboration diagram for _sble_state:



Data Fields

- uint8_t con

  The connection number. One uniqze number is assigned for each active connection.

- uint8_t addr [6]

  The remote's BLE address.

- uint8_t addr_type

  The address-type. Use either gap_address_type_random or ap_address_type_public.

- sble_ll * ll_rcvqueue

  Linked list for reception queue.

- sble_ll * ll_attributes

  Linked list for attribute lists (after a call to sble_attclient_getlist(), this list contains all attributes supported by the remote's GATT server)

4.7.1 Detailed Description

The sble_state struct is a per-conection-state and constitutes each entry in the cons[]-array.

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

- /home/kindt/workspace/HE2mT/projects/sble/include/sble_state.h

Chapter 5

File Documentation

5.1 /home/kindt/workspace/HE2mT/projects/sble/include/sble.h - File Reference

Master include for all sble-includes.

```
#include "sble_bgapi_call.h" #include "sble_io.h" #include
"sble_scheduler.h"   #include "sble_connect.h"   #include
"sble_types.h"  #include "sble_debug.h"  #include "sble_-
attclient.h"   #include "sble_event_handler_functions.h" ×
#include "sble_type_conversion.h"   #include "sble_gatt.h"
#include "sble_init.h"
```

5.1.1 Detailed Description

Master include for all sble-includes.

Date

09.07.2012

Author

: Philipp Kindt

5.2 /home/kindt/workspace/HE2mT/projects/sble/include/sble_-array.h File Reference

Datatype representing an array.

```
#include <stdlib.h> #include "sble_platform_config.h"
```

Data Structures

- struct _sble_array

  represents an array. Represents an array containing a pointer to allocated data and its size in bytes.

Typedefs

- typedef struct _sble_array **sble_array**

Functions

- void sble_array_malloc_data (sble_array *array, sble_unsigned_integer data_len)

  Allocate memory for the data of an array.
- void sble_array_free_data (sble_array *array)

  Free the space allocated to an sble_array's payload.
- void sble_array_malloc_whole (sble_array **array, sble_unsigned_integer data_len)

  Free the space allocated to an sble_array's payload and the sble_structure itself.
- void sble_array_free_whole (sble_array **array)

  Free the space allocated to an sble_array's payload and the corresponding sble_array-data-structure.
- sble_bool sble_array_comparator (void *a, void *b)

5.2.1 Detailed Description

Datatype representing an array. An array consists of a pointer to allocated data and its size in bytes.

Created on: 09.07.2012 Philipp Kindt <kindt@rcs.ei.tum.de>

5.2.2 Function Documentation

5.2.2.1 sble_bool sble_array_comparator (void * *a*, void * *b*)

Comparator function for sble_arrays. Returns true, if arrays *a and *b are equal. a,b: sble_arrays to compare

Returns

SBLE_TRUE, if both arrays have equal payload. SBLE_FALSE, if not.

5.2.2.2 void sble_array_free_data (sble_array * *array*)

Free the space allocated to an sble_array's payload.

Parameters

array	pointer to an sble_array structure. The sble_array for this pointer must be allocated before calling this function.

Returns

none

Examples:

sble_example_array.c.

5.2.2.3 void sble_array_free_whole (sble_array ** *array*)

Free the space allocated to an sble_array's payload and the corresponding sble_array-data-structure.

Parameters

array	pointer to a pointer to an sble_array structure. The sble_array for the *array - pointer must be allocated before calling this function.

Returns

none

Examples:

sble_example_array.c, and sble_example_client.c.

5.2.2.4 void sble_array_malloc_data (sble_array * *array*, sble_unsigned_integer *data_len*)

Allocate memory for the data of an array.

This function allocates memory for an sble_array's data structures, but not for the sble_array-structure itself.

Parameters

array	pointer to an sble_array structure that must allready be allocated
data_len	number of payload bytes in sble_array

Returns

none

Examples:

sble_example_array.c.

5.2.2.5 void sble_array_malloc_whole (sble_array ** *array*, sble_unsigned_integer *data_len*)

Free the space allocated to an sble_array's payload and the sble_structure itself.

Parameters

array	pointer to a pointer to an sble_array structure. The sble_array this pointer points to does not net to be allocated before. The *array-pointer will point to the newly allocated sble_array-structure after calling the function.
data_len	number of payload bytes in sble_array.

Returns

none

Examples:

sble_example_array.c, sble_example_client.c, and sble_example_minimal_client.c.

5.3 /home/kindt/workspace/HE2mT/projects/sble/include/sble_attclient.h File Reference

BLE attribute client (ATT) functions.

```
#include "sble_state.h"  #include "sble_platform_config.h" #include "sble_types.h" #include "sble_ll.h" #include "sble_bgapi_call.h" #include <inttypes.h>
```

Functions

- void sble_attclient_getlist (uint32_t con)
- sble_bool sble_attclient_is_in_list (uint32_t con, const sble_attribute *uuid)
- sble_attribute * sble_attclient_get_from_list (uint32_t con, const sble_attribute *att)
- sble_bool sble_attclient_write_by_handle (uint8_t con, uint16_t handle, sble_array *data)
- sble_bool sble_attclient_write_by_attribute (uint8_t con, sble_attribute *att, sble_array *data)

- sble_bool sble_attclient_write_by_uuid (uint8_t con, sble_array ∗uuid, sble_array ∗data)
- sble_payload ∗ sble_attclient_read_by_handle (uint8_t con, uint16_t handle)
- sble_payload ∗ sble_attclient_read_by_uuid (uint8_t con, sble_array ∗uuid)
- sble_payload ∗ sble_attclient_read_by_attribute (uint8_t con, sble_attribute ∗att)
- sble_payload ∗ sble_attclient_wait_for_payload (uint8_t con)

5.3.1 Detailed Description

BLE attribute client (ATT) functions. Attritbute protocol (ATT) client functions can be used to read and write attributes on a different node's GATT server. The there must be a BLE-connection established to the foreign host. Each attribute consists of a handle (which is an interger unique to each attribute on the atrribute server, but dependant of the node), a UUID (up to 16 bit value identifying an attribute uniquely, independant of the node) and its value.

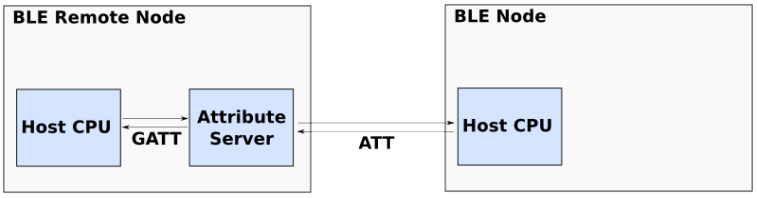


Figure 5.1: GATT and ATT

5.3.2 Function Documentation

5.3.2.1 sble_attribute∗ sble_attclient_get_from_list (uint32_t *con*, const sble_attribute ∗ *att*)

Returns a pointer on the entry in a connection's attribute list whose uuid corresponds to the one specified in the uuid-parameter.

Parameters

con	connection number
uuid	pointer to a sble_attribute-structure. The only field that has to be filled in the the uuid-param is uuid->uuid.

Returns

pointer to entry in the attribute-list dstate->cons[con].ll_gatt_values

Examples:

sble_example_client.c.

5.3.2.2 void sble_attclient_getlist (uint32_t *con*)

Retrives a list of attributes. It will be stored in dstate->cons[con].ll_gatt_values. If called more than once,the attribute list will be resetted before each call.

Parameters

con	connection number

Examples:

sble_example_check_for_attribute.c, sble_example_client.c, and sble_example_minimal_client.c.

5.3.2.3 sble_bool sble_attclient_is_in_list (uint32_t *con,* const sble_attribute * *uuid*)

Check if an attribute is within the attribute list dstate->cons[con].ll_gatt_values The list must have been retrived by sble_attclient_getlist() before.

Parameters

con	connection number

Examples:

sble_example_check_for_attribute.c, and sble_example_client.c.

5.3.2.4 sble_payload* sble_attclient_read_by_attribute (uint8_t *con,* sble_attribute * *att*)

Read an attribute on the remote node specified by con. The attribute is given by the attributes' global UUID.

Parameters

con	connection number
att	sble_attribute that contains the UUID of the attribute. All other fields of att are neglected

Returns

pointer to the payload received. This pointer should be freed when its content is not needed anymore by using sble_payload_free_whole()

5.3.2.5 sble_payload* sble_attclient_read_by_handle (uint8_t *con,* uint16_t *handle*)

Read an attribute on the remote node specified by con. The attribute is given by the handle on the remote server.

Parameters

con	connection number
handle	Unique number that qualifies the attribute on the remote node. It differs from node to node!

Returns

the payload (contains the handle's value)

5.3.2.6 sble_payload* sble_attclient_read_by_uuid (uint8_t *con*, sble_array * *uuid*)

Read an attribute on the remote node specified by con. The attribute is given by the attributes' global UUID.

Parameters

con	connection number
uuid	UUID of the attribute

Returns

pointer to the payload received. This pointer should be freed when its content is not needed anymore by using sble_payload_free_whole()

Examples:

sble_example_client.c.

5.3.2.7 sble_payload* sble_attclient_wait_for_payload (uint8_t *con*)

Waits until any payload is received from the node qualified by con.

Parameters

con	Connection number

Returns

pointer to the payload received. This pointer should be freed when its content is not needed anymore by using sble_payload_free_whole()

5.3.2.8 sble_bool sble_attclient_write_by_attribute (uint8_t *con*, sble_attribute * *att*, sble_array * *data*)

Write an attribute on the remote node specified by con. The attribute is given by the attributes' global UUID.

Parameters

con	connection number
att	sble_attribute that contains the UUID of the attribute. All other fields of att are neglected

Returns

SBLE_TRUE if attribute exists, SBLE_FALSE otherwise. No writing errors are detected currently.

Examples:

sble_example_client.c, and sble_example_minimal_client.c.

5.3.2.9 sble_bool sble_attclient_write_by_handle (uint8_t *con*, uint16_t *handle*, sble_array * *data*)

Write an attribute on the remote node specified by con. The attribute is given by the handle on the remote server.

Parameters

con	connection number
handle	Unique number that qualifies the attribute on the remote node. It differs from node to node!

Returns

SBLE_TRUE on success, SBLE_FALSE otherwise. At the moment, the return value is allways true.

5.3.2.10 sble_bool sble_attclient_write_by_uuid (uint8_t *con*, sble_array * *uuid*, sble_array * *data*)

Write an attribute on the remote node specified by con. The attribute is given by the attributes' global UUID.

Parameters

con	connection number
uuid	UUID of the attribute

Returns

SBLE_TRUE if attribute exists, SBLE_FALSE otherwise. No writing errors are detected currently.

5.4 /home/kindt/workspace/HE2mT/projects/sble/include/sble_attribute.h File Reference

Data structure representing an ATT/GATT attribute.

```
#include "sble_array.h"  #include "sble_platform_config.h" #include <inttypes.h>
```

Data Structures

- struct _sble_attribute

  Represents a GATT/ATT attribute.

Typedefs

- typedef struct _sble_attribute **sble_attribute**

Functions

- void sble_attribute_malloc_whole (sble_attribute **att, sble_unsigned_integer uuid_length)
- void sble_attribute_free_whole (sble_attribute **att)
- void sble_attribute_malloc_data (sble_attribute *att, sble_unsigned_integer uuid_length)
- void sble_attribute_free_data (sble_attribute *att)
- sble_bool sble_attribute_uuid_comparator (void *a, void *b)

5.4.1 Detailed Description

Data structure representing an ATT/GATT attribute.

Date

12.07.2012

Author

: Philipp Kindt

5.4.2 Function Documentation

5.4.2.1 void sble_attribute_free_data (sble_attribute * *att*)

Frees the uuid-data of an attribute

Parameters

att	pointer to an attribute

5.4.2.2 void sble_attribute_free_whole (sble_attribute ** *att*)

Frees the memory for an sble_attribute-structure and the uuid-data assigned to it.

Parameters

att	Pointer to an (initialized) pointer to a sble_attribute structure.

Examples:

sble_example_check_for_attribute.c, and sble_example_client.c.

5.4.2.3 void sble_attribute_malloc_data (sble_attribute * *att,* sble_unsigned_integer *uuid_length*)

Allocates data for the attribute's uuid

Parameters

att	pointer to an (allocated!) sble_attribute_structure. Its data-pointer will be assigned to newly allocated space.

5.4.2.4 void sble_attribute_malloc_whole (sble_attribute ** *att,* sble_unsigned_integer *uuid_length*)

Creates a new sble_attribute node. Give a pointer to an pointer to an sble_attribute. The sble_attribute will be created and the pointer is made pointing to it.

Parameters

att	Pointer to an unitialized pointer on an sble_attribute
uuid_lenght	Number of Bytes of the uuid of the attribute

Examples:

sble_example_check_for_attribute.c, sble_example_client.c, and sble_example_minimal_client.c.

5.4.2.5 sble_bool sble_attribute_uuid_comparator (void * *a,* void * *b*)

Compares two attributes. Returns true, if the attribute's UUIDs are equal.

Parameters

a,:	pointer to an sble_attribute whose uuid-field is set
b,:	pointer to an sble_attribute whose uuid-field is set

5.5 /home/kindt/workspace/HE2mT/projects/sble/include/sble_bgapi_call.h File Reference

Interface for blocking - and nonblocking calls to BGAPI BLE stack.

```
#include "sble_io.h"  #include "sble_state.h"  #include "sble_debug.h" #include <stdarg.h> #include "../bglib/cmd_def.h"
```

Defines

- #define sble_call_nb(...) ble_send_message(__VA_ARGS__);
- #define sble_call_bl(...) sble_bgapi_call_internal_bl_init(__VA_ARGS__);ble_send_message(__VA_ARGS__);sble_bgapi_call_internal_bl_delay();

Functions

- void sble_bgapi_call_internal_bl_init (uint8_t message,...)
- void sble_bgapi_call_internal_bl_delay ()

5.5.1 Detailed Description

Interface for blocking - and nonblocking calls to BGAPI BLE stack. -Nonblocking Call: An api command is sent and the calling function returns before the response has arrived.

WARNING: no BGAPI-Call must be sent until the previous command has been acknowledged by its response.

-Blocking Call: A BGAPI-Call is sent and the function returns to the callee after the response has been received.

Plase note a response is not the same as an event. For example: a ble_cmd_attributes_read is sent by the application. The BLE stack ackowledges this by a ble_rsp_attributes_read. When the data to be read arrives, a ble_evt_attclient_attribute_value event occurs.

Example: How to make a blocking call:

sble_call_bl(ble_cmd_attclient_find_information_idx,con,1,0xffff);

Parameters and conventions:

The parameters given to sble_call_bl() and sble_call_nb() are the same as for the BGAPI-Functions, except that the first param is the ble_msg_idx of the corresponding call. Example:

BGAPI : ble_cmd_attclient_find_information(con,1,0xffff);

SBLE call: sble_call_bl(ble_cmd_attclient_find_information_idx,con,1,0xffff);

Date

06.07.2012

Author

kindt

5.5.2 Define Documentation

5.5.2.1 #define sble_call_bl(...) sble_bgapi_call_internal_bl_init(__VA_ARGS__);ble_send_message(__VA_ARGS__);sble_bgapi_call_internal_bl_delay();

Perform a non-blocking call to an BGAPI function.

Parameters and conventions:

The parameters given to and sble_call_nb() are the same as for the BGAPI-Functions, except that the first param is the ble_msg_idx of the corresponding call.

Example:

BGAPI : ble_cmd_attclient_find_information(,con,1,0xffff);

SBLE call: sble_call_bl(ble_cmd_attclient_find_information_idx,con,1,0xffff);

Examples:

sble_example_client.c.

5.5.2.2 #define sble_call_nb(...) ble_send_message(__VA_ARGS__);

Issue a blocking call to the BGAPI.

Parameters and conventions:

The parameters given to sble_call_bl() are the same as for the BGAPI-Functions, except that the first param is the ble_msg_idx of the corresponding call.

Example:

BGAPI : ble_cmd_attclient_find_information(,con,1,0xffff);

SBLE call: sble_call_bl(ble_cmd_attclient_find_information_idx,con,1,0xffff);

5.5.3 Function Documentation

5.5.3.1 void sble_bgapi_call_internal_bl_delay ( )

Internal function used by BLE to wait for the response after a blocking call.

Never use.

5.5.3.2 void sble_bgapi_call_internal_bl_init (uint8_t *message,* ...)

Internal function used by BLE to prepare a blocking call to BGAPI

Never use.

5.6 /home/kindt/workspace/HE2mT/projects/sble/include/sble_connect.h File Reference

connection handling

```
#include "sble_bgapi_call.h" #include "sble_ll.h" #include <inttypes.h>
```

Enumerations

- enum **sble_addr_type** { **SBLE_ADDRESS_PUBLIC**, **SBLE_ADDRESS_RANDOM** }

Functions

- uint8_t sble_connect_to_any (uint16_t con_int_min, uint16_t con_int_max, uint16_t timeout, uint16_t latency)
- void sble_make_connectable_by_any (uint16_t adv_int_min, uint16_t adv_int_max)
- uint8_t sble_connect_to (sble_array *addr, sble_addr_type type, uint16_t con_interval_min, uint16_t con_interval_max, uint16_t timeout, uint16_t slave_latency)
- sble_bool sble_disconnect (uint8_t con)

5.6.1 Detailed Description

connection handling This file handles connections to external BLE nodes.

5.6.2 modes

A device can act in one of two roles:

Advertiser:

- Sends atvertising packets on channels 37,38 and 39
- Advertising packets can received by a device that is in scanner mode
- Advertising packets are sent repeatedly within an advertising interval **Scanner:**
- A Scanner listens for advertising packets
- In case of a reception of an adv packet, a "scan request" packet is sent
- The advertiser sents a scan response packet.

(Source and more information: LPRF San Diego, Bluetooth Low Energy Deep - Dive, http://e2e.ti.com/support/low_power_rf/m/videos__-files/653593/download.aspx)

To learn more about the exact timing, consult page 18 of: BLE Stack API reference v1.3, Bluegiga Technologies

Figure 5.2: advertiser and scanner

Date

06.07.2012

Author

kindt

5.6.3 Function Documentation

5.6.3.1 uint8_t sble_connect_to (sble_array * *addr,* sble_addr_type *type,* uint16_t *con_interval_min,* uint16_t *con_interval_max,* uint16_t *timeout,* uint16_t *slave_latency*)

Connect to the BLE-Node having a given BLE address. The function returns, if a connection has been established.

(Source for params: BLE Stack API reference v1.3, Bluegiga)

Parameters

addr	Address to connect to
con_int_min	Minimum connection interval (unit: 1.25ms), Range 7.5ms to 4000ms
con_int_max	Maximum connection Interval (unit: 1.25ms), Range 7.5ms to 4000ms
timeout	Supervision timeout (unit: 10ms) - Range 100ms to 32 seconds, must be bigger than connection interval. If no packets are received either by the master or the slave within this time interval, the connection is terminated.
latency	Slave latency - the number of connection intervals the slave need not response. Higher latency will save energy. Range: 0-500

Returns

Connection number.

5.6.3.2 uint8_t sble_connect_to_any (uint16_t *con_int_min*, uint16_t *con_int_max*, uint16_t *timeout*, uint16_t *latency*)

Connect to any BLE-Node in range by starting scanning and connecting to the first scan response that has been received. The function returns, if a connection has been established. (Source for params: BLE Stack API reference v1.3, Bluegiga)

Parameters

con_int_min	Minimum connection interval (unit: 1.25ms), Range 7.5ms to 4000ms
con_int_max	Maximum connection Interval (unit: 1.25ms), Range 7.5ms to 4000ms
timeout	Supervision timeout (unit: 10ms) - Range 100ms to 32 seconds, must be bigger than connection interval. If no packets are received either by the master or the slave within this time interval, the connection is terminated.
latency	Slave latency - the number of connection intervals the slave need not response. Higher latency will save energy. Range: 0-500

Returns

Connection number.

Examples:

sble_example_check_for_attribute.c, sble_example_client.c, and sble_example_minimal_client.c.

5.6.3.3 sble_bool sble_disconnect (uint8_t *con*)

Close a connection.

Parameters

con	Connection number to terminate

Returns

SBLE_TRUE on success, SBLE_FALSE otherwise. Not all errors are deteceted, yet.

Examples:

sble_example_check_for_attribute.c.

5.6.3.4 void sble_make_connectable_by_any (uint16_t *adv_int_min*, uint16_t *adv_int_max*)

Make the device connectable by any device. After calling this function, it returns immediately but the BLE-Radio will start advertising. To wait for a connection to be established, use sble_scheduler_wait_for_event() and trigger for SBLE_STATE_CONNECTION_EVENT.

(Source for params: BLE Stack API reference v1.3, Bluegiga)

Parameters

adv_int_min	Minimum advertisment interval (unit: 625us)
adv_int_max	Maximum advertisment interval (unit: 625us)

Examples:

sble_example_gattserver.c, and sble_example_minimal_gattserver.c.

5.7 /home/kindt/workspace/HE2mT/projects/sble/include/sble_debug.h File Reference

Debugging tools for SBLE.

```
#include "sble_platform_config.h" #include "FreeRTOS.h"
```

Defines

- #define SBLE_DEBUG(...) printf("[DEBUG: %s,l. %d, %s @ %s] ",__FILE__, __LINE__, __FUNCTION__, pcTaskGetTaskName(NULL)); printf(__VA_ARGS__); printf("\n");
- #define SBLE_DEBUG_CON(...) printf(__VA_ARGS__);
- #define SBLE_ERROR(...) printf("[ERROR: %s,l. %d, %s @ %s] ",__FILE__, __LINE__,__FUNCTION__,pcTaskGetTaskName(NULL)); printf(__VA_ARGS__); printf("\n"); vTaskEndScheduler(); hard_fault_handler();

- #define SBLE_ERROR_CONTINUABLE(...) printf("[ERROR_CONTINUABLE: %s,l. %d] ",__FILE__, __LINE__); printf(__VA_ARGS__); printf("\n");

Functions

- void sble_print_hex_array (const uint8_t *data, uint32_t len)
- void sble_print_char_array (const uint8_t *data, uint32_t len)
- void sble_print_bitfield (sble_unsigned_integer field, sble_unsigned_integer length)

  Prints the bit values (0 or 1) of given bitfield.
- void print_backtrace ()

  Prints a backtrace of the current call-situation to stdout.

5.7.1 Detailed Description

Debugging tools for SBLE. Debugging tools to write on stdout.

Date

06.07.2012

Author

Philipp Kindt

5.7.2 Define Documentation

5.7.2.1 #define SBLE_DEBUG(...) printf("[DEBUG: %s,l. %d, %s @ %s] ",__FILE__, __LINE__, __FUNCTION__, pcTaskGetTaskName(NULL)); printf(__VA_ARGS__); printf("\n");

Print a debug message to STDOUT. File, Line and Function are printed in addition to the debug message itself. Syntax: The same as for printf(). It will terminate the message with a newline automatically.

Examples:

sble_example_check_for_attribute.c, sble_example_client.c, sble_example_gattserver.c, sble_example_ll.c, sble_example_minimal_client.c, and sble_example_minimal_gattserver.c.

5.7.2.2 #define SBLE_DEBUG_CON(...) printf(__VA_ARGS__);

Print a debug message to STDOUT. File, Line and Function are not printed in addition to the debug message itself. Syntax: The same as for printf(). It won't terminate the message with a newline automatically.

Examples:

sble_example_client.c, sble_example_gattserver.c, and sble_example_minimal_gattserver.c.

5.7.2.3 #define SBLE_ERROR(...) printf("[ERROR: %s,l. %d, %s @ %s] ",__FILE__, __LINE__,__FUNCTION__,pcTaskGetTaskName(NULL)); printf(__VA_ARGS__); printf("\n"); vTaskEndScheduler(); hard_fault_handler();

Print a rrror message to STDOUT and terminate the program. File, Line and Function are printed in addition to the error message itself. Syntax: The same as for printf(). It will terminate the message with a newline automatically.

5.7.2.4 #define SBLE_ERROR_CONTINUABLE(...) printf("[ERROR_CONTINUABLE: %s,l. %d] ",__FILE__, __LINE__); printf(__VA_ARGS__); printf("\n");

Print a rrror message to STDOUT, but do not terminate the program. File, Line and Function are printed in addition to the error message itself. Syntax: The same as for printf(). It will terminate the message with a newline automatically.

5.7.3 Function Documentation

5.7.3.1 void print_backtrace ( )

Prints a backtrace of the current call-situation to stdout.

Make sure that the linker-flag -rdynamic is set, otherwise yo will just get raw addresses instead of symbols. This code is taken from the Linux-kernel manpages @ http://www.kernel.org/doc/man-pages/ and has been slightly modified by - Philipp Kindt <kindt@rcs.ei.tum.de>

Original copyright notice: Copyrights: These man pages come under various copyrights. All pages are freely distributable when the nroff source is included.

5.7.3.2 void sble_print_bitfield (sble_unsigned_integer *field*, sble_unsigned_integer *length*)

Prints the bit values (0 or 1) of given bitfield.

Parameters

field	The value to print
lenght	the numer of bits to print. Max. sizeof(field)!

5.7.3.3 void sble_print_char_array (const uint8_t * *data*, uint32_t *len*)

Print the data of an array as their asci-chars to stdout

Parameters

data	Pointer to data to print
len	Number of bytes to print - max. the lenght of the data-array

Examples:

sble_example_gattserver.c.

5.7.3.4 void sble_print_hex_array (const uint8_t * *data*, uint32_t *len*)

Print the data of an array as hex-string to stdout

Parameters

data	Pointer to data to print
len	Number of bytes to print - max. the lenght of the data-array

Examples:

sble_example_client.c, sble_example_gattserver.c, and sble_example_minimal_gattserver.c.

5.8 /home/kindt/workspace/HE2mT/projects/sble/include/sble_event_handler_functions.h File Reference

functions internally used by SBLE & BGAPI as handler functions for some events.

```
#include "../bglib/cmd_def.h"
```

Functions

- void sble_evth_connection_established (const struct ble_msg_connection_status_evt_t *msg)
- void sble_evth_disconnected (const struct ble_msg_connection_disconnected_evt_t *msg)

5.8.1 Detailed Description

functions internally used by SBLE & BGAPI as handler functions for some events. Never use theese functions. They are used by the callbacks in bglib/commands.c

Date

11.07.2012

Author

Philipp Kindt

5.8.2 Function Documentation

5.8.2.1 void sble_evth_connection_established (const struct ble_msg_connection_status_evt_t * *msg*)

This function is called autmatically if a connection has been established. It will set up the correpsonding queues and more. Never use directly.

5.8.2.2 void sble_evth_disconnected (const struct ble_msg_connection_disconnected_evt_t * *msg*)

This function is called autmatically if a connection has been terminateed. It free the correpsonding queues and more. Never use directly.

5.9 /home/kindt/workspace/HE2mT/projects/sble/include/sble_gatt.h File Reference

Functions to access the local attribute (GATT) server.

```
#include "sble_types.h" #include "sble_scheduler.h" #include
"sble_bgapi_call.h"
```

Functions

- void sble_gatt_write_by_handle (uint16_t handle, sble_array *data)
- sble_payload * sble_gatt_recieve ()
- sble_payload * sble_gatt_read_by_handle (uint16_t handle)
- sble_array * sble_gatt_get_type (uint16_t handle)

5.9.1 Detailed Description

Functions to access the local attribute (GATT) server. The local attribute server stores the values for attributes that can be read and/or written by external nodes using the ATT-Protocol.

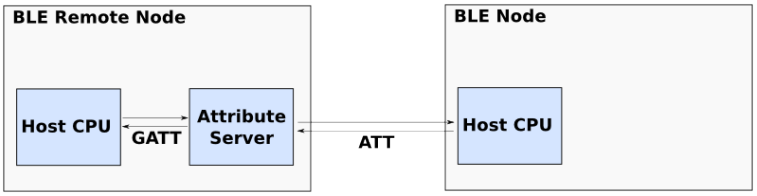


Figure 5.3: GATT and ATT

Date

12.07.2012

Author

Philipp Kindt

5.9.2 Function Documentation

5.9.2.1 sble_array* sble_gatt_get_type (uint16_t *handle*)

Read out the UUID of an attribute qualified by its handle

This value is not pushed to the GATT receive queue (dstate.pl_gatt_value) as it receives fully deterministically only due to a corresponding request. It is copied to dstate.pl_gatt_value in struct _sble_driver_state(). Do never free the value returned as the pointer points to dstate.pl_gat_value. It will automatically freed when the next value is requested, so do not double-free. If you need more than one of theese values at the same time, make sure to copy the old data somewhere else.

Parameters

handle	Number identifying the attribute on the device

Returns

Pointer to the payload containing to the attribute's value.

Parameters

handle	Number identifying the attribute on the device

Returns

UUID of the attribute

5.9.2.2 sble_payload* sble_gatt_read_by_handle (uint16_t *handle*)

Read the value of an attribute qualified by its handle.

This value is not pushed to the GATT receive queue (dstate.pl_gatt_value) as it receives fully deterministically only due to a corresponding request. It is copied to dstate.pl_gatt_value in struct _sble_driver_state(). Do never free the value returned as the pointer points to dstate.pl_gat_value. It will automatically freed when the next value is requested, so do not double-free. If you need more than one of theese values at the same time, make sure to copy the old data somewhere else.

Parameters

handle	Number identifying the attribute on the device

Returns

Pointer to the payload containing to the attribute's value.

Examples:

sble_example_gattserver.c.

5.9.2.3 sble_payload* sble_gatt_recieve ( )

Wait until a connected device modifies any value on the local GATT server. Any data received is pushed onto the gatt's receive queue (dstate.ll_gatt_values). A pointer to the element added most recently in this queue is returned and the list-entry is removed. Make sure to free the sble_payload structure afterwards using sble_payload_free_whole() .

Returns

Pointer to the payload received.

Examples:

sble_example_gattserver.c, and sble_example_minimal_gattserver.c.

5.9.2.4 void sble_gatt_write_by_handle (uint16_t *handle,* sble_array * *data*)

Write attribute by its local handle. The handle is unique for one device/firmware, but not among different devices having different firmwares

Parameters

handle	Number identifying the attribute on the device
data	The data the attribute's value will be set to.

Examples:

sble_example_gattserver.c.

5.10 /home/kindt/workspace/HE2mT/projects/sble/include/sble_init.h File Reference

Initialization and shutdown of SBLE.

```
#include <inttypes.h>
```

Functions

- void sble_init (uint8_t *param)
- void sble_shutdown ()

5.10.1 Detailed Description

Initialization and shutdown of SBLE.

Date

16.07.2012

Author

: kindt

5.10.2 Function Documentation

5.10.2.1 void sble_init (uint8_t * *param*)

Initialize SBLE

Starts the callback dispatcher thread and initializes all data structures. Tries to connect to the BLE112 device.

Examples:

sble_example_check_for_attribute.c, sble_example_client.c, sble_example_gattserver.c, sble_example_minimal_client.c, and sble_example_minimal_gattserver.c.

5.10.2.2 void sble_shutdown ( )

Shut down SBLE

Stops the callback dispatcher thread and disconnects the I/O to the BLE112 device.

Examples:

sble_example_check_for_attribute.c, sble_example_client.c, and sble_example_-gattserver.c.

5.11 /home/kindt/workspace/HE2mT/projects/sble/include/sble_-io.h File Reference

Input/Output via Serial UART to BLE112-Device.

```
#include <inttypes.h> #include "sble_platform_config.h"
```

Functions

- void sble_io_init (char *param)
- void sble_io_disconnect ()
- void sble_io_out (uint8_t len1, uint8_t *dbuf1, uint16_t len2, uint8_t *dbuf2)
- sble_signed_integer sble_io_read ()
- void sble_io_reset (char *device)

5.11.1 Detailed Description

Input/Output via Serial UART to BLE112-Device. This file does the I/O to the BLE112 or BLED112-Device. The header-file is platform-independant, whereas the corresponding .c-files is highly platform-dependant. It must be reimplemented for any platform. Naming Convention for implementations of this header: sble_io_[platform].c

Author

Philipp Kindt

Date

06.2012

5.11.2 Function Documentation

5.11.2.1 void sble_io_disconnect ( )

Disconnects from blutooth low energy module

5.11.2.2 void sble_io_init (char * *param*)

Establihes a connection to bluetooth low energy module via a (pseudo-)serial device node such as /dev/ttyACM0 on posix

Parameters

param	A parameter. Its interpretation is plattform-dependant. On posix: path to the device file connecting to the radio, such as /dev/ttyACM0

5.11.2.3 void sble_io_out (uint8_t *len1*, uint8_t * *dbuf1*, uint16_t *len2*, uint8_t * *dbuf2*)

Send BGAPI-message to bluetooth low energy module via BGAPI

Parameters

len1	length of first part of message (header)
dbuf1	data for first part of the message (header)
len2	length of second part of the message (payload)
dbuf2	data of the second part of the message (payload)

5.11.2.4 sble_signed_integer sble_io_read ( )

Read out BGAPI-message. This function must be polled continuously. It invokes the callbacks for BGAPI

5.11.2.5 void sble_io_reset (char * *device*)

reset BLE112/BLED112 device.

Parameters

device	Parameter qualifiing the device. Its interpretation is plattform-dependant. On posix: path to the device file connecting to the radio, such as /dev/ttyACM0

5.12 /home/kindt/workspace/HE2mT/projects/sble/include/sble_ll.h File Reference

Double-Linked list.

```
#include <inttypes.h>   #include "sble_platform_config.h"
#include "sble_debug.h"
```

Data Structures

- struct _sble_ll_node

  A node of the linked list.
- struct _sble_ll

  The structure representing a linked list.

Typedefs

- typedef sble_bool(∗ sble_ll_comparator_fct)(void ∗, void ∗)
- typedef struct _sble_ll_node sble_ll_node

  A node of the linked list.
- typedef struct _sble_ll sble_ll

  The structure representing a linked list.

Functions

- void sble_ll_init (sble_ll ∗ll)
- sble_ll_node ∗ sble_ll_find_last_iterating (sble_ll ∗ll)
- sble_bool sble_ll_isempty (sble_ll ∗ll)
- sble_bool sble_ll_push (sble_ll ∗ll, void ∗data)
- sble_bool sble_ll_push_unique (sble_ll ∗ll, void ∗data, sble_ll_comparator_fct isEqual)
- void ∗ sble_ll_pop (sble_ll ∗ll)
- sble_ll_node ∗ sble_ll_get_element (sble_ll ∗ll, sble_ll_comparator_fct isEqual, void ∗data)
- void ∗ sble_ll_get_and_remove_element (sble_ll ∗ll, sble_ll_comparator_fct isEqual, void ∗data)
- void sble_ll_remove_all_equal_to (sble_ll ∗ll, sble_ll_comparator_fct isEqual, void ∗data)
- void sble_ll_free_nodes (sble_ll ∗ll, sble_bool free_data)
- sble_unsigned_integer sble_ll_get_nr_of_elements (sble_ll ∗ll)
- sble_ll_node ∗ sble_ll_get_next (sble_ll_node ∗node)

5.12.1 Detailed Description

Double-Linked list. sble_ll.h implements a simple, general-purpose double-linked list. It is used for the reception queues.

Date

06.07.2012

Author

: kindt

5.12.2 Typedef Documentation

5.12.2.1 typedef sble_bool(∗ sble_ll_comparator_fct)(void ∗, void ∗)

A function returning SBLE_TRUE if both elements are equal and SBLE_FALSE otherwise.

5.12.3 Function Documentation

5.12.3.1 sble_ll_node∗ sble_ll_find_last_iterating (sble_ll ∗ *ll*)

Find the last node of a linked list by iterating from the root to the list's end. Normally, this function need not be called. use the last-pointer in sble_ll.h first.

Parameters

ll	Pointer to the linked list.

Returns

Pointer to a sble_node or NULL in case of failure

5.12.3.2 void sble_ll_free_nodes (sble_ll ∗ *ll,* sble_bool *free_data*)

Free all nodes of a list.

Parameters

ll	Pointer to the linked list.
free_data	if SBLE_TRUE, the elements of the list are free'ed to. Otherwise, only the sble_node-structures are freed. All list pointers are adjusted accordingly.

5.12.3.3 void∗ sble_ll_get_and_remove_element (sble_ll ∗ *ll,* sble_ll_comparator_fct *isEqual,* void ∗ *data*)

Returns a pointer to an element which es equal to another element according to a comparator function. Equality is defined by the comparator function. Thus, the elements can be scanned for a certain property. The element found will be removed from the list.

Parameters

ll	Pointer to the linked list.
data	Pointer to data to compare the list-elements with
isEqual	Comparator function to check weather two elements are equal or not

Returns

Pointer to a sble_node or NULL in case of failure

5.12.3.4 sble_ll_node* sble_ll_get_element (sble_ll * *ll*, sble_ll_comparator_fct *isEqual*, void * *data*)

Returns a pointer to an element which es equal to another element according to a comparator function. Equality is defined by the comparator function. Thus, the elements can be scanned for a certain property. The element found will remain in list.

Parameters

ll	Pointer to the linked list.
data	Pointer to data to compare the list-elements with
isEqual	Comparator function to check weather two elements are equal or not

Returns

Pointer to a sble_node or NULL in case of failure

5.12.3.5 sble_ll_node* sble_ll_get_next (sble_ll_node * *node*)

Get next node in a linked list.

Parameters

Node	A node whose next node shall be determined

Returns

Pointer to the next node in list or NULL if no node could befound.

Examples:

sble_example_client.c.

5.12.3.6 sble_unsigned_integer sble_ll_get_nr_of_elements (sble_ll * *ll*)

Return the number of elements in a list.

Parameters

ll	Pointer to the linked list.

5.12.3.7 void sble_ll_init (sble_ll * *ll*)

Initialize a linked list

Parameters

ll	pointer to the linked list. The pointer must allready point to an alloc'ed linked list.

Examples:

sble_example_ll.c.

5.12.3.8 sble_bool sble_ll_isempty (sble_ll * *ll*)

Check if linked list is empty.

Parameters

ll	Pointer to the linked list.

Returns

SBLE_TRUE if emtpy, SBLE_FALSE if not.

5.12.3.9 void* sble_ll_pop (sble_ll * *ll*)

Pop the element added most recently from the linked list

Parameters

ll	Pointer to the linked list.

Returns

SBLE_TRUE in case of success or SBLE_FALSE in case of failure

Examples:

sble_example_ll.c.

5.12.3.10 sble_bool sble_ll_push (sble_ll * *ll*, void * *data*)

Push a value on top of the linked list.

Parameters

ll	Pointer to the linked list.
data	Pointer to data to push onto linked list

Returns

SBLE_TRUE in case of success or SBLE_FALSE in case of failure

Examples:

sble_example_ll.c.

5.12.3.11 sble_bool sble_ll_push_unique (sble_ll * *ll*, void * *data*, sble_ll_comparator_fct *isEqual*)

Push a value on top of the linked list and replace the old one, if the entry already exist.

Parameters

ll	Pointer to the linked list.
data	Pointer to data to push onto linked list
isEqual	Comparator function to check weather two elements are equal or not

Returns

SBLE_TRUE in case of success or SBLE_FALSE in case of failure

5.12.3.12 void sble_ll_remove_all_equal_to (sble_ll * *ll*, sble_ll_comparator_fct *isEqual*, void * *data*)

Remove all elements that are equal to a given element according to a comparator function. Equality is defined by the comparator function. Thus, all elements having a certain property can be removed.

Parameters

ll	Pointer to the linked list.
data	Pointer to data to compare the list-elements with
isEqual	Comparator function to check weather two elements are equal or not

5.13 /home/kindt/workspace/HE2mT/projects/sble/include/sble_payload.h File Reference

Data structure representing payload.

`#include <inttypes.h>` `#include "sble_array.h"`

Data Structures

- struct _sble_payload

  Structure representing payload that is transmitted via ATT protocol.

Defines

- #define sble_payload_get_data(ppl) return(ppl->data)

Typedefs

- typedef struct _sble_payload sble_payload

  Structure representing payload that is transmitted via ATT protocol.

Functions

- void sble_payload_malloc_whole (sble_payload **pl, sble_unsigned_integer data_length)
- void sble_payload_free_whole (sble_payload **pl)
- void **sble_payload_malloc_data** (sble_payload *pl, sble_unsigned_integer data_length)
- void **sble_payload_free_data** (sble_payload *pl)

5.13.1 Detailed Description

Data structure representing payload. Data structure containing a data buffer, an attribute handle and a connection number representing payload transmitted via the BLE radio.

Date

06.07.2012

Author

Philipp Kindt

5.13.2 Define Documentation

5.13.2.1 #define sble_payload_get_data(*ppl*) return(ppl->data)

Get data buffer from payload. Not Typesafe!

Parameters

pl	pointer on a sble_payload structure

Returns

pointer to sble_array that contains the payload's data

5.13.3 Function Documentation

5.13.3.1 void sble_payload_free_whole (sble_payload ** *pl*)

Free a sble_payload structure. Frees both the data and the sble_payload-structure itself

Parameters

pl,:	Pointer to a pointer to a sble_payload_structure. *pl must point to sble_payload-structure createdby sble_payload_malloc_whole(). *pl will be set to NULL.

5.13.3.2 void sble_payload_malloc_whole (sble_payload ** *pl,* sble_unsigned_integer *data_length*)

Create a new sble_payload structure.

Parameters

pl,:	Pointer to a pointer to a sble_payload_structure. *pl should be unitialized. The alloc'ed memory will be assigned to pl.
data_lenght	Number of payload bytes - i.e. the number of bytes for the value to be transmitted.

5.14 /home/kindt/workspace/HE2mT/projects/sble/include/sble_platform_config.h File Reference

Plattform-dependant configuration.

```
#include <inttypes.h>
```

Defines

- #define SBLE_PLATTFORM_ARCHITECTURE_POSIX 1

  an ident for the posix-plattform
- #define **SBLE_PLATTFORM_ARCHITECTURE_STM32F4_FREERTOS** 2
- #define SBLE_BUF_MAXLEN 50

- #define SBLE_FASTMODE 1

  if defined, the code is optimized for speed at the cost of Debuggability.
- #define SBLE_TRUE 1

  Values for sble_bool.
- #define **SBLE_FALSE** 0

Typedefs

- typedef uint32_t sble_bool

  A general-purpose boolean value. Plattform-dependant.
- typedef uint32_t sble_unsigned_integer

  A unsigned integer. Its size if platform-dependant.
- typedef int32_t sble_signed_integer

  A signed integer. Its size if platform-dependant.

5.14.1 Detailed Description

Plattform-dependant configuration. This file must be adjusted (among some others) if SBLE is beeing ported to a different platform. It contains platform-secific configruation values.

Date

06.07.2012

Author

: kindt

5.14.2 Define Documentation

5.14.2.1 #define SBLE_BUF_MAXLEN 50

The actual plattform architecture possible values:

- SBLE_PLATTFORM_ARCHITECTURE_POSIX => Unix/Posix (ISO/IEEE 9945)
- SBLE_PLATTFORM_ARCHITECTURE_STM32F4_FREERTOS => FreeRTOS on stm32f4 providing read/write I/O to ble module The maximum lenght for all reception buffers. Memory will only be allocated if they're really filled that much.

5.15 /home/kindt/workspace/HE2mT/projects/sble/include/sble_scheduler.h File Reference

Scheduling for sble. Plattform-dependant!

```
#include <inttypes.h> #include "sble_platform_config.h"
```

Typedefs

- typedef enum _sble_thread sble_thread

Enumerations

- enum _sble_thread { **SBLE_THREAD_MAIN**, **SBLE_THREAD_DISPATCHER** }

Functions

- void sble_scheduler_init ()
- void * sble_callback_dispatcher (void *threadarg)
- void sble_scheduler_wait (sble_thread thread)
- void sble_scheduler_unlock_mutex ()
- void sble_scheduler_lock_mutex ()
- void sble_scheduler_wakeup (sble_thread thread)
- void sble_scheduler_wait_for_event (sble_thread thread, uint32_t event_flag_mask)
- void sble_scheduler_wait_for_event_no_reset (sble_thread thread, uint32_t event_flag_mask)
- void sble_scheduler_dispatcher_shutdown (sble_thread thread)
- void sble_scheduler_dispatcher_start ()
- void sble_scheduler_autoclear_prevent (uint32_t events)
- void sble_scheduler_autoclear_do (uint32_t events)
- void sble_scheduler_events_clear (uint32_t events)
- void sble_scheduler_events_set (uint32_t events)

5.15.1 Detailed Description

Scheduling for sble. Plattform-dependant!

5.15.2 in SBLE

This files contains everything related to the scheduling. SBLE runs within two threads:

- The main thread which receives function calls from the programm using SBLE
- The callback dispatcher thread which is woken up by the operating system every time data is received via the UART.

5.15.2.1 events

The BLE radio can signal two message types:

- Responses to a command issued by the main thread before
- Asynchronous events (example: Value received via radio link)

Everytime something from the BLE radio is received, the callback dispatcher thread is woken up by the operating system. Depending on the preceeding calls within the main thread, the callback dispatcher perfoms different actions. In any case, a callback corresponding to the event or response that occured is called. Within the callback, the global state variable dstate is modified usually. In some cases, received vallues are pushed onto one of the reception stack.

5.15.2.2 calls

Most SBLE-functionallity is handled by sble-function-calls from the application using S-BLE. These calls invoke the SBLE-Scheduler with different possibilities **Non-Blocking call:**

- The main thread sends an api command and returns to caller immediately.
- The callback dispatcher thread is not invoked

Blocking call:

- The main thread sends an api command and goes asleep
- The dispatcher thread wakes up the main thread if the response for this call has been signaled
- The dispatcher thread goes asleep
- the main thread wakes up the dispatcher thread if it has received the response to the command
- the calling functions returns and the dispatcher thread sleeps until the next data is received from the radio

Event waiting functions

Some functions use void sble_scheduler_wait_for_event(). This function waits until at least one event within specified set of events had occured. It works like this:

- First, a blocking call is issued (see above) callback dispatcher thread sleeps until any event occures callback dispatcher thread signals the event to the main thread and wakes it up and goes asleep after that main thread checks if the event it has ben waiting for occured. If yes, it wakes up the dispatcher thread that will go asleep again to be woken up by the operating system and returns to the caller. If not, it wakes up the dispatcher thread that will wait for the oerating system again, but will set the main-thread asleep again until the next event occurs.

Date

06.07.2012

Author

: Philipp Kindt

5.15.3 Typedef Documentation

5.15.3.1 typedef enum _sble_thread sble_thread

Enum that specifies one of two threads:

- SBLE_THREAD_MAIN the main thread
- SBLE_THREAD_DISPATCHER the callback dispatcher thread

5.15.4 Enumeration Type Documentation

5.15.4.1 enum _sble_thread

Enum that specifies one of two threads:

- SBLE_THREAD_MAIN the main thread
- SBLE_THREAD_DISPATCHER the callback dispatcher thread

5.15.5 Function Documentation

5.15.5.1 void* sble_callback_dispatcher (void * *threadarg*)

The callback dispatcher thread function. It contains an endless loop and is to be started as its own thread by sble_scheduler_init(). It will call the callbacks and, to a huge extend, handle the sleep-and-wakeup-procedures for the main thread in collaboration with other functions

Parameters

threadarg	Argument for the thread - unused.

5.15.5.2 void sble_scheduler_autoclear_do (uint32_t *events*)

Add the events specified by a given bitfield to the event autoclear list. All events in this list will be acknowledged autmatically, so the dispatcher thread continues automatically if one of these events occur. If an event occurs that is not on this list, the dispatcher thread will sleep until woken up by another thread.

Parameters

events	Bitfield specifying the events to remove from the aec list. It can conatin any combination of the SBLE_STATE_∗-macros defined in sble_state.h

5.15.5.3 void sble_scheduler_autoclear_prevent (uint32_t *events*)

Removes the events specified by a given bitfield fromthe event autoclear list. All events in this list will be acknowledged autmatically, so the dispatcher thread continues automatically if one of these events occur. If an event occurs that is not on this list, the dispatcher thread will sleep until woken up by another thread.

Parameters

events	Bitfield specifying the events to remove from the aec list. It can conatin any combination of the SBLE_STATE_∗-macros defined in sble_state.h

5.15.5.4 void sble_scheduler_dispatcher_shutdown (sble_thread *thread*)

Shuts down the event dispatcher thread.

Parameters

thread	Even though it is possible to shut down the main thread, too, this param should allways be SBLE_THREAD_DISPATCHER. Usually called by sble_shutdown().

5.15.5.5 void sble_scheduler_dispatcher_start ( )

Start the dispatcher thead. Usually called by sble_scheduler_init() and, thus, sble_init();

5.15.5.6 void sble_scheduler_events_clear (uint32_t *events*)

After an event has occured, for most events, a flag is set in the global state's flags (dstate.flags) that indicates the event has occured. This function clears the events that might have occured that are specified by a bitmask

Parameters

events	bitmask specifiing the events to be cleared

5.15.5.7 void sble_scheduler_events_set (uint32_t *events*)

After an event has occured, for most events, a flag is set in the global state's flags (dstate.flags) that indicates the event has occured. This function makrks some events specified by a bitmask as "occured"

Parameters

events	bitmask specifiing the events to be set in the global state's (dstate.flags)

5.15.5.8 void sble_scheduler_init ( )

Initialize the scheduler and start the dispatcher thread

5.15.5.9 void sble_scheduler_lock_mutex ( )

Lock a synchronisation mutex to access global shared data.

5.15.5.10 void sble_scheduler_unlock_mutex ( )

Unlock a synchronisation mutex to access global shared data.

5.15.5.11 void sble_scheduler_wait (sble_thread *thread*)

Make a thread sleep.

Parameters

thread	The thread-identifier qualifying the thread that shall go asleep..

5.15.5.12 void sble_scheduler_wait_for_event (sble_thread *thread,* uint32_t *event_flag_mask*)

Wait for one event out of a list of given events to occur. Until this occured, set the thread of the calling function asleep.

Procedure:

- The function calling this function must first remove all events of flag_mask from the event autoclear list in the global state (dstate.evt_clear_list) by using sble_scheduler_autoclear_prevent
- Usually, the calling function issues a BGAPI-command
- After that, sble_scheduler_wakeup() is called. It waits until one of the events specified occures

- sble_scheduler_wakeup() restores those events to the event autoclear list.

Parameters

thread	The thread identifier of the calling thread
event_flag_mask	A bitfiled specifying the events to wait for. This can be any combination of the SBLE_STATE_*-macros defined in sble_state.h

5.15.5.13 void sble_scheduler_wait_for_event_no_reset (sble_thread *thread*, uint32_t *event_flag_mask*)

Wait for one event out of a list of given events to occur. Until this occured, set the thread of the calling function asleep.

Same as sble_scheduler_wait_for_event(), but does not restore the event autoclear list automatically. Procedure:

- The function calling this function must first remove all events of flag_mask from the event autoclear list in the global state (dstate.evt_clear_list) by using sble_scheduler_autoclear_prevent
- Usually, the calling function issues a BGAPI-command
- After that, sble_scheduler_wakeup() is called. It waits until one of the events specified occures

Parameters

thread	The thread identifier of the calling thread
event_flag_mask	A bitfiled specifying the events to wait for. This can be any combination of the SBLE_STATE_*-macros defined in sble_state.h

5.15.5.14 void sble_scheduler_wakeup (sble_thread *thread*)

Wake up a sleeping thread.

Parameters

thread	The thread-identifier qualifying the calling thread.

5.16 /home/kindt/workspace/HE2mT/projects/sble/include/sble_state.h File Reference

The state of SBLE.

```
#include "sble_platform_config.h"  #include "sble_ll.h" ×
#include "sble_types.h"  #include "FreeRTOS.h"  #include
"semphr.h" #include "task.h"
```

Data Structures

- struct _sble_state

  The sble_state struct is a per-conection-state and constitutes each entry in the cons[]-array.
- struct _sble_driver_state

  The global state (dstate-Variable) of SBLE.

Defines

- #define SBLE_STATE_CMD_SUCCESS 1

  a response to a command has been answered successfully
- #define SBLE_STATE_TERMINATE 2

  the system is about to terminate/is currently terminating
- #define SBLE_STATE_CMD_SENT 4

  a command has been sent
- #define SBLE_STATE_RESPONSE_RECEIVED 8

  the response of a command has been recieved
- #define SBLE_STATE_EVENT 16

  an arbitrary event has occured
- #define SBLE_STATE_SCAN_RESPONSE_EVENT 32

  a scan response received
- #define SBLE_STATE_CONNECTION_EVENT 64

  connection has been established
- #define SBLE_STATE_ATTRIBUTE_INFORMATION_FOUND_EVENT 128

  a scan for attributes has revealed an attribute
- #define SBLE_STATE_ATTRIBUTE_PROCEDURE_COMPLETED_EVENT 256

  ATT procedure has been completed. This might happen after for example attribute information lists are transferred.
- #define SBLE_STATE_ATTRIBUTE_VALUE_EVENT 512

  an attribute value has been received
- #define SBLE_STATE_GATT_VALUE_RESPONSE 1024

  a GATT value has been read by request from the local gatt db
- #define SBLE_STATE_GATT_VALUE_EVENT 2048

  a GATT value has been received from remote node via ATT
- #define SBLE_EVENT_CLEAR_LIST_DEFAULT (∼(0) & ∼(SBLE_STATE_TERMINATE|SBLE_STATE_CMD_SENT|SBLE_STATE_RESPONSE_RECEIVED|SBLE_STATE_EVENT))

  The default event clear list. Clears everything but SBLE_STATE_TERMINATE|SBLE_STATE_CMD_SENT|SBLE_STATE_RESPONSE_RECEIVED|SBLE_STATE_EVENT autmatically.
- #define SBLE_FLAG_SCHED_FOR_RESPONSE_ACK 1

  Internal flag for the scheduler that a command response has been acknowledged.

Typedefs

- typedef void(∗ sble_event_handler)()
- typedef struct _sble_state sble_state

  The sble_state struct is a per-conection-state and constitutes each entry in the cons[]-array.
- typedef struct _sble_driver_state sble_driver_state

  The global state (dstate-Variable) of SBLE.

Functions

- void sble_state_init ()
- void sble_state_finalize ()

Variables

- sble_driver_state **dstate**

5.16.1 Detailed Description

The state of SBLE. The state of BLE consists of a global variable of type sble_driver_-state, called dstate. It contains properties that are global to all connections, and at its cons[]-array, properties that are individual to each acitve connection. This File is highly plattform dependant and has to be adapted if SBLE is to be ported to a new plattform.

Date

06.07.2012

Author

kindt

5.16.2 Typedef Documentation

5.16.2.1 typedef void(∗ sble_event_handler)()

A handler function for events. Currently unused.

5.16.3 Function Documentation

5.16.3.1 void sble_state_finalize ( )

Finalizes the global state. Destroys all the linked-lists used by sble_state abd sble_-driver_state.

5.16.3.2 void sble_state_init ( )

Initialitzes the global state. Creates all the linked-lists used.

5.17 /home/kindt/workspace/HE2mT/projects/sble/include/sble_type_conversion.h File Reference

Utilities to convert one state to another.

```
#include "sble_types.h"
```

Functions

- sble_array * sble_type_conversion_hexstring_to_binary (char *str_hex)
- uint8_t sble_type_conversion_char_2_numeric (char ch)

5.17.1 Detailed Description

Utilities to convert one state to another. This function can convert hex strings to sble_arrays.

Date

12.07.2012

Author

: Philipp Kindt

5.17.2 Function Documentation

5.17.2.1 uint8_t sble_type_conversion_char_2_numeric (char *ch*)

helper function used by ble_type_conversion_hexstring_to_binary(). Returns the numeric value of a asci char (for example. 'c' sird zu

5.17.2.2 sble_array* sble_type_conversion_hexstring_to_binary (char * *str_hex*)

convert a hex string into a sble_array.

Parameters

str_hex	An hex string such as "ABcdeFG1234";

Returns

The sble_array that is returned is newly created and should be free'ed after use by sble_array_free_whole().

Examples:

sble_example_client.c, and sble_example_gattserver.c.

5.18 /home/kindt/workspace/HE2mT/projects/sble/include/sble_types.h File Reference

Data Types master include.

```
#include "sble_payload.h"    #include "sble_attribute.h" ×
#include "sble_array.h"   #include "sble_ll.h"   #include
"sble_platform_config.h"
```

5.18.1 Detailed Description

Data Types master include. This file includes other header files of SBLE that contain many data-types.

Date

09.07.2012

Author

: Philipp Kindt

Chapter 6

Example Documentation

6.1 sble_example_array.c

```
#include "sble_array.h"


int main(){
        sble_array* arr;
        sble_array_malloc_whole(&arr,5);                        //allocate
       space for 5 bytes
         arr->data[0] = 24;
         arr->data[1] = 12;
         arr->data[2] = 15;                                     //... some
       payload
         sble_array_free_whole(&arr);

         sble_array arr_allready_malloced;              //this structure is
       allready in memory => just allocate the payload

         sble_array_malloc_data(&arr_allready_malloced,5);
         //do something nifty with the data
         sble_array_free_data(&arr_allready_malloced);

         return 0;
}
```

6.2 sble_example_check_for_attribute.c

```
#include "sble.h"
#include "sble_debug.h"


int main(){
        sble_attribute* att;
        sble_init("/dev/ttyACM0");
        uint8_t con = sble_connect_to_any(40,40);               //connect to
       any node in range.

        sble_attribute_malloc_whole(&att,2);                    //make space
```

```
        for an attribute having a 2 bytes UUID
         att->uuid->data[1] = 0x28;
         att->uuid->data[0] = 0x03;

         //retrive attribute list for connection 0
         sble_attclient_getlist(con);

         //check if attribute exist at remote
         if(sble_attclient_is_in_list(con,att)){
                 SBLE_DEBUG("Attribute found.");
         }else{
                 SBLE_DEBUG("Attribute not found.");
         }

         //tidy up...
         sble_attribute_free_whole(&att);
         sble_disconnect(con);
         sble_shutdown();

          return 0;
}
```

6.3 sble_example_client.c

```
#include "sble.h"
#include "../bglib/cmd_def.h"
#include <inttypes.h>
#include <string.h>             //for memcpy

int main(){
        uint32_t cnt;
        sble_array* data;
        sble_attribute* att;

        //define our attribute's UUIDs. This would have been eaiser with
       ble_type_conversion_hexstring_to_binary(), but just to show you another way...
        uint8_t uuid_rd[16] = {0xa1,0xba,0xd3,0x99,0x94,0x42,0x0d,0x8e,0x7a,
      0x4e,0x33,0x03,0x40,0xb8,0x95,0xd1};
        uint8_t uuid_wr[16] = {0x53,0x0a,0xcc,0xd9,0xe7,0xd4,0x90,0xb9,0x57,
      0x41,0x0a,0xc5,0xe5,0x2c,0x65,0x6a};

        sble_init("/dev/ttyACM1");

        SBLE_DEBUG("Connecting to any node in range...");
        sble_connect_to_any(40,40,400,0);

        //Or, if you want to connect to a specific node
        //
      sble_connect_to(sble_type_conversion_hexstring_to_binary("dc934c800700"), SBLE_ADDRESS_

        SBLE_DEBUG("Connected. Retriving attribute list...");
        //retrive the attribute list at the remote's GATT server
        sble_attclient_getlist(0);
        SBLE_DEBUG("retrived list.");



        //create an attribute and check if this uuid is in attribute list
        sble_attribute_malloc_whole(&att,2);
```

```
 att->uuid->data[1] = 0x28;
 att->uuid->data[0] = 0x03;
 if(sble_attclient_is_in_list(0,att)){
        SBLE_DEBUG("Attribute exists at remote.");
 }else{
        SBLE_DEBUG("Attribute does not exist at the remote.");
 }
 sble_attribute_free_whole(&att);               //free our "test
attribute"




 //Now lets print out the whole attribute list
 sble_ll_node* n;
 n = (sble_ll_node*) dstate.cons[0]->ll_attributes->root;
 while((n = sble_ll_get_next(n)) != NULL){
        att = ((sble_attribute*) n->data);
        SBLE_DEBUG_CON("Atthandle %d: ",att->handle);
        sble_print_hex_array(att->uuid->data,att->uuid->len);
 };




 //now write the value "knorke" to the remote's gatt server, to
attribute uuid_wr

 //-> create an attribute type to write
 sble_attribute_malloc_whole(&att,16);
 memcpy(att->uuid->data,uuid_wr,16);

 //and create the data to write
 sble_array_malloc_whole(&data,5);
 memcpy(data->data,"knorke",5);

 //do it and tidy up
 sble_attclient_write_by_attribute(0,att,data);
 sble_array_free_whole(&data);




 //now read out uuid_rd. If the sble_example_gattserver.c - demo is
running on the remote,
 //this will be our knorke...

 sble_payload *pl;

 //generate uuid...
 sble_array* uuid;
 uuid = sble_type_conversion_hexstring_to_binary("
a1bad39994420d8e7a4e330340b895d1");
```

```
        SBLE_DEBUG_CON("Converted UUID :");
        sble_print_hex_array(uuid->data,uuid->len);

        //read out
        pl = sble_attclient_read_by_uuid(0,uuid);

        //tidy up
        sble_array_free_whole(&uuid);
        if(pl != NULL){
                SBLE_DEBUG_CON("Data received (main): ");
                sble_print_hex_array(pl->data->data,pl->data->len);
        }




        //now we want to pick one particular attribute from the list

        //prepare attribute structure to retrice
        sble_attribute* atr;
        sble_attribute_malloc_whole(&att,16);
        memcpy(att->uuid->data,uuid_rd,16);
        att->uuid->len=16;

        //retrive from list
        atr = sble_attclient_get_from_list(0,att);

        if(atr != NULL){
                SBLE_DEBUG_CON("attribute found");
        }else{
                SBLE_DEBUG("atr is NULL - not reading attribute");
        }
        sble_attribute_free_whole(&att);


        //disconnect after a few seconds
        sleep(10);
        sble_call_bl(ble_cmd_connection_disconnect_idx,0);
        sleep(1);
        sble_shutdown();
        return 0;
}
```

6.4 sble_example_gattserver.c

```
#include "sble.h"

int main(){
        sble_init("/dev/ttyACM0");

        //make the device connectable by any node who wishes to
        sble_make_connectable_by_any(400,400);


        //create an array having the data 0xcaffee1234
        sble_array* arr = sble_type_conversion_hexstring_to_binary("caffee1234"
      );
```

```
        sble_print_hex_array(arr->data,arr->len);

        //write data array to gatt server
        sble_gatt_write_by_handle(20,arr);

        //read out what we have written
        sble_payload* pl = sble_gatt_read_by_handle(20);
        if(pl != NULL){
                SBLE_DEBUG_CON("H20 read from GATT server: ");
                sble_print_hex_array(pl->data->data,pl->data->len);
        }

        //now read out handle 16
        pl = sble_gatt_read_by_handle(16);
        if(pl != NULL){
                SBLE_DEBUG_CON("H16 read from GATT server: ");
                sble_print_char_array(pl->data->data,pl->data->len);

        }

        //now wait for a client writing to the GATT server via ATT
        SBLE_DEBUG("waiting for incoming transfer.");
        pl = sble_gatt_recieve();


        if(pl != NULL){
                SBLE_DEBUG_CON("read from remote node: ");
                sble_print_char_array(pl->data->data,pl->data->len);

        }else{
                SBLE_DEBUG("No Payload received.");
        }


        //Wait for some time and shutdown
        sleep(10);
        sble_shutdown();
        return 0;
}
```

6.5 sble_example_ll.c

```
#include "sble_ll.h"
#include "sble_debug.h"

int main(){
        //create linked list
        sble_ll ll;
        sble_ll_init(&ll);

        int a,b,c,d;
        a = 1;
        b = 2;
        c = 3;
        d = 4;

        //fill list
        sble_ll_push(&ll,&a);
        sble_ll_push(&ll,&b);
        sble_ll_push(&ll,&c);
```

```
        sble_ll_push(&ll,&d);


        //... and pop values from list
        int* value;
        while(value = sble_ll_pop(&ll)){
                SBLE_DEBUG("Pop: %d",*value);
        }
        return 0;
}
```

6.6 sble_example_minimal_client.c

Date

: 09.07.2012

Author

Philipp Kindt

```
#include "sble.h"
#include "../bglib/cmd_def.h"
#include <inttypes.h>
#include <string.h>             //for memcpy

int main(){
        uint32_t cnt;
        sble_array* data;
        sble_attribute* att;

        //define our attribute's UUIs. This would have been eaiser with
       ble_type_conversion_hexstring_to_binary(), but just to show you another way...
        uint8_t uuid_wr[16] = {0x53,0x0a,0xcc,0xd9,0xe7,0xd4,0x90,0xb9,0x57,
      0x41,0x0a,0xc5,0xe5,0x2c,0x65,0x6a};
        sble_init("/dev/ttyACM1");

        SBLE_DEBUG("Connecting to any node in range...");
        sble_connect_to_any(40,40,400,0);


        SBLE_DEBUG("connection established!");







        //now write the value "test12" to the remote's gatt server, to
       attribute uuid_wr

        //-> create an attribute type to write
        sble_attribute_malloc_whole(&att,16);           //UUID has 16 byte
        memcpy(att->uuid->data,uuid_wr,16);

        //and create the data to write
        sble_array_malloc_whole(&data,5);                       //5 bytes to
       write
```

```
        uint8_t buf[5] = {0xca, 0xff, 0xee,0x12,0x34};  //"caffee1234"
        memcpy(data->data, buf,5);                      //write value

        sble_attclient_getlist(0);

        //do it!
        while(1){
                SBLE_DEBUG("writing attribute...");
                sble_attclient_write_by_attribute(0,att,data);

                //wait some time
                cnt = 0;
                while(cnt < 168000000ul){
                        cnt++;
                }
        }



        //never reached
        return 0;
}
```

6.7 sble_example_minimal_gattserver.c

```
#include "sble.h"

int main(){
        sble_init("/dev/ttyACM0");

        //make the device connectable by any node who wishes to
        sble_make_connectable_by_any(400,400);



        //now wait for a client writing to the GATT server via ATT
        SBLE_DEBUG("waiting for incoming transfer.");

        sble_payload* pl;
        while(1){
                pl = sble_gatt_recieve();
                if(pl != NULL){
                        SBLE_DEBUG_CON("read from remote node: ");
                        sble_print_hex_array(pl->data->data,pl->data->len);

                }else{
                        SBLE_DEBUG("No Payload received.");
                }
        }

        //never reached
        return 0;
}
```