## Contents

1 BLEeMod: A precise energy model for the Bluetooth Low Energy Protocol  
   1.1 Credits: .............................................................. 1  
   1.2 Introduction .......................................................... 1  
   1.3 Compiling .............................................................. 1  
   1.4 Porting ................................................................. 1  
   1.5 License ................................................................. 1  

2 Data Structure Index  
   2.1 Data Structures ........................................................... 3  

3 File Index  
   3.1 File List ................................................................. 5  

4 Data Structure Documentation  
   4.1 _discoveryModelResult_t Struct Reference  
       4.1.1 Detailed Description ........................................... 7  

5 File Documentation  
   5.1 src/ble_model.h File Reference  
       5.1.1 Detailed Description ........................................... 9  
   5.2 src/ble_model_connected.h File Reference  
       5.2.1 Detailed Description ........................................... 10  
       5.2.2 Function Documentation  
           5.2.2.1 ble_e_model_c_getChargeConnectionInterval ........... 11  
           5.2.2.2 ble_e_model_c_getChargeConnectionIntervalSamePayload 12  
           5.2.2.3 ble_e_model_c_getChargeConstantParts ................. 12  
           5.2.2.4 ble_e_model_c_getChargeEvent .......................... 13  
           5.2.2.5 ble_e_model_c_getChargeEventSamePayload ............ 13  
           5.2.2.6 ble_e_model_c_getChargeSequences ..................... 13  
           5.2.2.7 ble_e_model_c_getChargeSequencesSamePayload ........ 14  
           5.2.2.8 ble_e_model_c_getDurationConstantParts ............. 14  
           5.2.2.9 ble_e_model_c_getDurationEvent ....................... 15  
           5.2.2.10 ble_e_model_c_getDurationEventSamePayload ......... 16
Chapter 1

BLEeMod: A precise energy model for the Bluetooth Low Energy Protocol

1.1 Credits:

Power Model: 2013, Philipp Kindt kindt@rcs.ei.tum.de 2013, Daniel Yunge yunge@rcs.ei.tum.de 2013, Robert Diemer diemer@rcs.ei.tum.de 2013, Samarjit Chakraborty samarjit@tum.de

This Software: 2013, Philipp Kindt kindt@rcs.ei.tum.de

1.2 Introduction

This library contains an easy-to-use and precise energy model for the BLE Protocol. The Data Included is valid for BLE112-devices. The data can be changed for other devices. How to do this: Please refer to the publication "A precise Energy Model for the Bluetooth Low Energy Protocol".

All energies are presented in the units of electric charge [As]. By multiplying these values with the supply voltage of the BLE module, Energy in [J] is obtained.

1.3 Compiling

The library compiles on Linux, libmath from the GNU C library (or any other compatible c-library providing math.h) must be linked ("-lm" - flag). To compile the examples, go to the examples directory and type "make".

1.4 Porting

This programm compiles under linux/posix only, if the BLE_MODEL_PLATTFORM macro is set to BLE_MODEL_- PLATTFORM_POSIX. In this case won't compile on embedded platforms, but can easily ported: When there is an error, it calls printf() exit(). This is the only platform-dependand call. setBLE_MODEL_PLATTFORM to BLE_MO- DEL_PLATTFORM_OTHER, and it will compile on any plattform that comes with a c-compiler! Disadvantage: No error messages are issued. The software does not need malloc().

1.5 License

(c) 2013, Philipp Kindt (c) 2013, Lehrstuhl für Realzeit-Computersysteme (RCS), Technische Universität München (TUM)
This file is part of bleemod.

bleemod is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

bleemod is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with bleemod. If not, see http://www.gnu.org/licenses/
Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

`_discoveryModelResult_t` ................................................................. 7
Chapter 3

File Index

3.1 File List

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

- `src/ble_model.h`
  Master-include for the BLE Power model ........................................... 9
- `src/ble_model_connected.h`
  BLE Energy model for the connected mode ........................................... 10
- `src/ble_model_connection_establishment.h`
  Energy model for BLE connection request procedures and for connection update procedures . 17
- `src/ble_model_discovery.h`
  Energy model for device discovery in BLE ........................................... 19
- `src/ble_model_params.h`
  Master include for all numerical values .............................................. 23
- `src/ble_model_params_connected.h`
  Device-dependant model params for Bluegiga BLE112 devices in connected mode ........ 23
- `src/ble_model_params_connection_establishment.h`
  Energy model params for connection establishment and connection parameter updates ........ 25
- `src/ble_model_params_general.h`
  Device-dependant model params for Bluegiga BLE112 devices that are independant from the mode (connected/advertising/scanning/...) ........................................... 26
- `src/ble_model_params_scanning.h`
  Model parameters for scan events for BLE112-devices .............................. 27
- `src/ble_model_scanning.h`
  Energy model for BLE scan events .................................................. 28
Chapter 4

Data Structure Documentation

4.1  _discoveryModelResult_t Struct Reference

#include <ble_model_discovery.h>

Data Fields

• double discoveryLatency
  
  The discovery latency [s].

• double chargeAdv
  
  The charge consumed by the advertiser for device discovery [As].

• double chargeScan
  
  The charge consumed by the scanner for device discovery [As].

4.1.1 Detailed Description

Result of the model for device discovery.

Examples:

  examples.c.

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

• src/ble_model_discovery.h
Chapter 5

File Documentation

5.1 src/ble_model.h File Reference

Master-include for the BLE Power model.

```
#include "ble_model_connected.h"
#include "ble_model_scanning.h"
#include "ble_model_connection_establishment.h"
#include "ble_model_discovery.h"
```

Macros

- `#define BLE_MODEL_PLATTFORM_POSIX 1`
  
  possible value for the `BLE_MODEL_PLATTFORM` : Posix/Linux operating systems. Only advantage when enabling this: The program will abort with exit() in case of an error

- `#define BLE_MODEL_PLATTFORM_OTHER 2`
  
  possible value for the `BLE_MODEL_PLATTFORM` : any other plattform

- `#define BLE_MODEL_PLATTFORM BLE_MODEL_PLATTFORM_POSIX`
  
  can be set to `BLE_MODEL_PLATTFORM_POSIX` if it is going to be compiled for Linux/Posix. Otherwise, change to any different value to compile without posix libraries. Only disadvantage: Programm does not abort with exit() in case of error messages.

5.1.1 Detailed Description

Master-include for the BLE Power model.

- Main include file for Energy Model for the Bluetooth Low Energy Protocol

Date

: 1.10.2013

Author

Philipp Heinrich Kindt
5.2 src/ble_model_connected.h File Reference

BLE Energy model for the connected mode.

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

### Macros

- `#define BLE_E_MODE_INT_MAXSEQUENCES 15`

  maximum number of communication sequences possible (for pre-mallocing to avoid the use of malloc)

### Functions

- `double ble_e_model_c_getChargeConstantParts ()`
- `double ble_e_model_c_getDurationConstantParts ()`
- `double ble_e_model_c_getChargeSequences (uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t n-Rx[], uint8_t nTx[], uint8_t txPower)`
- `double ble_e_model_c_getChargeSequencesSamePayload (uint8_t masterOrSlave, double Tc, uint8_t n-Seq, uint8_t n-Rx[], uint8_t nTx[], uint8_t txPower)`
- `double ble_e_model_c_getDurationSequences (uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t n-Rx[], uint8_t nTx[], uint8_t txPower)`
- `double ble_e_model_c_getDurationSequencesSamePayload (uint8_t masterOrSlave, double Tc, uint8_t n-Seq, uint8_t nRx, uint8_t nTx, uint8_t txPower)`
- `double ble_e_model_c_getChargeEvent (uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t n-Rx[], uint8_t nTx[], uint8_t txPower)`
- `double ble_e_model_c_getDurationEvent (uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t n-Rx[], uint8_t nTx[], uint8_t txPower)`
- `double ble_e_model_c_getChargeEventSamePayload (uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t nRx, uint8_t nTx, uint8_t txPower)`
- `double ble_e_model_c_getDurationEventSamePayload (uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t nRx, uint8_t nTx, uint8_t txPower)`
- `double ble_e_model_c_getChargeConnectionInterval (uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t n-Rx[], uint8_t nTn.Tx[], uint8_t txPower)`
- `double ble_e_model_c_getChargeConnectionIntervalSamePayload (uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t n-Rx[], uint8_t nTx[], uint8_t txPower)`

5.2.1 Detailed Description

BLE Energy model for the connected mode. (c) 2013, Philipp Kindt (c) 2013, Lehrstuhl für Realzeit-
Computersysteme (RCS), Technische Universität München (TUM)

This file is part of bleemod.

bleemod is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

bleemod is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with bleemod. If not, see http://www.gnu.org/licenses/
5.2 Function Documentation

5.2.2.1 double ble_e_model_c_getChargeConnectionInterval ( uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t nRx[], uint8_t nTx[], uint8_t txPower )

Returns the charge consumed within a BLE connection interval. It includes both the connection event and the sleep duration. Each sequence may have an unique number of bytes sent and received.
### 5.2.2.2 double ble_e_model_c_getChargeConnectionIntervalSamePayload ( uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t nRx, uint8_t nTx, uint8_t txPower )

Returns the charge consumed within a BLE connection interval. It includes both the connection event and the sleep duration. Each sequence must have the same number of bytes sent and received.

**Parameters**

- **masterOrSlave:** 1 = master, 0 = slave. For the slave, Rx and Tx are swapped (Master: first Tx, then Rx; Slave: first Rx, then Tx) and there is window-widening and a longer dPreRx for the first sequence in an event.
- **Tc** Connection interval
- **nSeq** Number of sequences (pairs of packets per connection event)
- **nRx[]** Number of bytes received. Each array element contains the number of bytes received per sequence (pair of packet). Must include all protocol overheads.
- **nTx[]** Number of bytes sent. Each array element contains the number of bytes sent per sequence (pair of packet). Must include all protocol overheads.
- **txPower** Tx-Power setting of the device

**Returns**

Charge consumed within one connection interval [C]

### Examples:

`examples.c`

#### 5.2.2.3 double ble_e_model_c_getChargeConstantParts ( )

Returns the charge of all constant parts of the model. These are: head, preprocessing, transient state, postprocessing, tail

**Returns**

charge of all constant parts [C]

**Examples:**

`examples.c`
5.2.2.4 double ble_e_model_c_getChargeEvent ( uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t nRx[], uint8_t nTx[], uint8_t txPower )

Returns the charge consumed by a BLE connection event. Each sequence may have a unique number of bytes sent and received. The sleep duration in the connection interval is not accounted for. See ble_e_model_c_getChargeConnectionInterval() for the charge consumed per interval.

Parameters

<table>
<thead>
<tr>
<th>masterOrSlave:</th>
<th>1=&gt;master, 0=&gt;Slave. For the slave, Rx and Tx are swapped (Master: first Tx, then Rx; Slave: first Rx, then Tx) and there is window-widening and a longer dPreRx for the first sequence in an event.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tc</td>
<td>Connection interval</td>
</tr>
<tr>
<td>nSeq</td>
<td>Number of sequences (pairs of packets per connection event)</td>
</tr>
<tr>
<td>nRx[]</td>
<td>Number of bytes received. Each array element contains the number of bytes received per sequence (pair of packet). Must include all protocol overheads.</td>
</tr>
<tr>
<td>nTx[]</td>
<td>Number of bytes sent. Each array element contains the number of bytes sent per sequence (pair of packet). Must include all protocol overheads.</td>
</tr>
<tr>
<td>txPower</td>
<td>Tx-Power setting of the device</td>
</tr>
</tbody>
</table>

Returns

Charge consumed by the connection event [C]

5.2.2.5 double ble_e_model_c_getChargeEventSamePayload ( uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t nRx, uint8_t nTx, uint8_t txPower )

Returns the charge consumed by a BLE connection event. Each sequence must have the same number of bytes sent and received. The sleep duration in the connection interval is not accounted for. See ble_e_model_c_getChargeConnectionIntervalSamePayload() for the charge consumed per interval.

Parameters

<table>
<thead>
<tr>
<th>masterOrSlave:</th>
<th>1=&gt;master, 0=&gt;Slave. For the slave, Rx and Tx are swapped (Master: first Tx, then Rx; Slave: first Rx, then Tx) and there is window-widening and a longer dPreRx for the first sequence in an event.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tc</td>
<td>Connection interval</td>
</tr>
<tr>
<td>nSeq</td>
<td>Number of sequences (pairs of packets per connection event)</td>
</tr>
<tr>
<td>nRx</td>
<td>Number of bytes received. Each array element contains the number of bytes received per sequence (pair of packet). Must include all protocol overheads.</td>
</tr>
<tr>
<td>nTx</td>
<td>Number of bytes sent. Each array element contains the number of bytes sent per sequence (pair of packet). Must include all protocol overheads.</td>
</tr>
<tr>
<td>txPower</td>
<td>Tx-Power setting of the device</td>
</tr>
</tbody>
</table>

Returns

Charge consumed by the connection event [C]

Examples:

examples.c.

5.2.2.6 double ble_e_model_c_getChargeSequences ( uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t nRx[], uint8_t nTx[], uint8_t txPower )

Returns the charge of the communication sequence phases. Each sequence may have a unique number of bytes sent and received. These are: Communication preamble, Window-Widening (slave), Rx, Rx2Tx, Tx, Tx2Rx
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>masterOrSlave,</td>
<td>1=&gt;master, 0=&gt;Slave. For the slave, Rx and Tx are swapped (Master: first Tx, then Rx; Slave: first Rx, then Tx) and there is window-widening and a longer dPreRx for the first sequence in an event.</td>
</tr>
<tr>
<td>Tc</td>
<td>Connection interval</td>
</tr>
<tr>
<td>nSeq</td>
<td>Number of sequences (pairs of packets per connection event)</td>
</tr>
<tr>
<td>nRx[]</td>
<td>Number of bytes received. Each array element contains the number of bytes received per sequence (pair of packet). Must include all protocol overheads.</td>
</tr>
<tr>
<td>nTx[]</td>
<td>Number of bytes sent. Each array element contains the number of bytes sent per sequence (pair of packet). Must include all protocol overheads.</td>
</tr>
<tr>
<td>txPower</td>
<td>Tx-Power setting of the device</td>
</tr>
</tbody>
</table>

Returns

Charge consumed by the sequences [C]

5.2.2.7  double ble_e_model_c_getChargeSequencesSamePayload ( uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t nRx, uint8_t nTx, uint8_t txPower )

Returns the charge of the communication sequence phases. Each must have the same number of bytes sent or received (to overcome this limitation, please use ble_e_model_c_getChargeSequences() ) These sequences are: Communication preamble, Window-Widening (slave), Rx, Rx2Tx, Tx, Tx2Rx

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>masterOrSlave,</td>
<td>1=&gt;master, 0=&gt;Slave. For the slave, Rx and Tx are swapped (Master: first Tx, then Rx; Slave: first Rx, then Tx) and there is window-widening and a longer dPreRx for the first sequence in an event.</td>
</tr>
<tr>
<td>Tc</td>
<td>Connection interval</td>
</tr>
<tr>
<td>nSeq</td>
<td>Number of sequences (pairs of packets per connection event)</td>
</tr>
<tr>
<td>nRx[]</td>
<td>Number of bytes received. Each array element contains the number of bytes received per sequence (pair of packet). Must include all protocol overheads.</td>
</tr>
<tr>
<td>nTx[]</td>
<td>Number of bytes sent. Each array element contains the number of bytes sent per sequence (pair of packet). Must include all protocol overheads.</td>
</tr>
<tr>
<td>txPower</td>
<td>Tx-Power setting of the device</td>
</tr>
</tbody>
</table>

Returns

Charge consumed by the sequences [C]

Examples:

examples.c

5.2.2.8  double ble_e_model_c_getDurationConstantParts ( )

Returns the duration of all constant parts of the model. These are: head, preprocessing, transient state, postprocessing, tail

Returns

duration of all constant parts [t]

Examples:

examples.c
5.2.2.9 double ble_e_model_c_getDurationEvent ( uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t nRx[], uint8_t nTx[], uint8_t txPower )

Returns the duration of a BLE connection event. Each sequence may have an unique number of bytes sent and received. The sleep duration in the connection interval is not accounted for.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>masterOrSlave</td>
<td>1=&gt;master, 0=&gt;Slave. For the slave, Rx and Tx are swapped (Master: first Tx, then Rx; Slave: first Rx, then Tx) and there is window-widening and a longer dPreRx for the first sequence in an event.</td>
</tr>
<tr>
<td>Tc</td>
<td>Connection interval</td>
</tr>
<tr>
<td>nSeq</td>
<td>Number of sequences (pairs of packets per connection event)</td>
</tr>
<tr>
<td>nRx[]</td>
<td>Number of bytes received. Each array element contains the number of bytes received per sequence (pair of packet). Must include all protocol overheads.</td>
</tr>
<tr>
<td>nTx[]</td>
<td>Number of bytes sent. Each array element contains the number of bytes sent per sequence (pair of packet). Must include all protocol overheads.</td>
</tr>
<tr>
<td>txPower</td>
<td>Tx-Power setting of the device</td>
</tr>
</tbody>
</table>

Returns

Duration of the connection event [s]

5.2.2.10 double ble_e_model_c_getDurationEventSamePayload ( uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t nRx, uint8_t nTx, uint8_t txPower )

Returns the duration of a BLE connection event. Each sequence must have the same number of bytes sent and received. The sleep duration in the connection interval is not accounted for.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>masterOrSlave</td>
<td>1=&gt;master, 0=&gt;Slave. For the slave, Rx and Tx are swapped (Master: first Tx, then Rx; Slave: first Rx, then Tx) and there is window-widening and a longer dPreRx for the first sequence in an event.</td>
</tr>
<tr>
<td>Tc</td>
<td>Connection interval</td>
</tr>
<tr>
<td>nSeq</td>
<td>Number of sequences (pairs of packets per connection event)</td>
</tr>
<tr>
<td>nRx[]</td>
<td>Number of bytes received. Each array element contains the number of bytes received per sequence (pair of packet). Must include all protocol overheads.</td>
</tr>
<tr>
<td>nTx[]</td>
<td>Number of bytes sent. Each array element contains the number of bytes sent per sequence (pair of packet). Must include all protocol overheads.</td>
</tr>
<tr>
<td>txPower</td>
<td>Tx-Power setting of the device</td>
</tr>
</tbody>
</table>

Returns

Duration of the connection event [s]

Examples:

examples.c.

5.2.2.11 double ble_e_model_c_getDurationSequences ( uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t nRx[], uint8_t nTx[], uint8_t txPower )

Returns the duration of the communication sequence phases. Each sequence may have a unique number of bytes sent and received. Sequences accounted for are: Communication preamble, Window-Widening (slave), Rx, Rx2Tx, Tx, Tx2Rx

Parameters
5.3 src/ble_model_connection_establishment.h File Reference

| masterOrSlave: | 1=>master, 0=>Slave. For the slave, Rx and Tx are swapped (Master: first Tx, then Rx; Slave: first Rx, then Tx) and there is window-widening and a longer dPreRx for the first sequence in an event. |
| Tc | Connection interval |
| nSeq | Number of sequences (pairs of packets per connection event) |
| nRx[] | Number of bytes received. Each array element contains the number of bytes received per sequence (pair of packet). Must include all protocol overheads. |
| nTx[] | Number of bytes sent. Each array element contains the number of bytes sent per sequence (pair of packet). Must include all protocol overheads. |
| txPower | Tx-Power setting of the device |

Returns

Time taken by the communication sequences [s]

5.2.2.12 double ble_e_model_c_getDurationSequencesSamePayload ( uint8_t masterOrSlave, double Tc, uint8_t nSeq, uint8_t nRx, uint8_t nTx, uint8_t txPower )

Returns the time spent by the communication sequence phases. Each must have the same number of bytes sent or received (to overcome this limitation, please use ble_e_model_c_getDurationSequences() ) These sequences are: Communication preamble, Window-Widening (slave), Rx, Rx2Tx,Tx,Tx2Rx

Parameters

| masterOrSlave: | 1=>master, 0=>Slave. For the slave, Rx and Tx are swapped (Master: first Tx, then Rx; Slave: first Rx, then Tx) and there is window-widening and a longer dPreRx for the first sequence in an event. |
| Tc | Connection interval |
| nSeq | Number of sequences (pairs of packets per connection event) |
| nRx[] | Number of bytes received. Each array element contains the number of bytes received per sequence (pair of packet). Must include all protocol overheads. |
| nTx[] | Number of bytes sent. Each array element contains the number of bytes sent per sequence (pair of packet). Must include all protocol overheads. |
| txPower | Tx-Power setting of the device |

Returns

Time spent within the communication sequences [s]

Examples:

examples.c.

5.3 src/ble_model_connection_establishment.h File Reference

Energy model for BLE connection request procedures and for connection update procedures.

#include <inttypes.h>
#include "ble_model_params_connection_establishment.h"
#include "ble_model_scanning.h"

Functions

- double ble_e_model_ce_getChargeForConnectionProcedure (uint8_t establishmentOrUpdate, ble_e_model_sc_scan_type scanType, uint8_t masterOrSlave, double TcOld, double TcNew)
Computes the charge consumed by establishing a connection or updating the connection parameters of an existing connection.

### 5.3.1 Detailed Description

Energy model for BLE connection request procedures and for connection update procedures. This file implements a model for connection parameter update procedures and for connection request procedures.

2.10.2013, Philipp Kindt kindt@rcs.ei.tum.de

(c) 2013, Philipp Kindt (c) 2013, Lehrstuhl für Realzeit-Computersysteme (RCS), Technische Universität München (TUM)

This file is part of bleemod.

bleemod is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

bleemod is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with bleemod. If not, see http://www.gnu.org/licenses/

### 5.3.2 Function Documentation

5.3.2.1 double ble_e_model_ce_getChargeForConnectionProcedure ( uint8_t establishmentOrUpdate, ble_e_model_sc_scan_type scanType, uint8_t masterOrSlave, double TcOld, double TcNew )

Computes the charge consumed by establishing a connection or updating the connection parameters of an existing connection.

This function computes the charge consumed by a connection request procedure or connection parameter update procedure. A connection request procedure occurs, after a scanner has received an ADV_IND advertising packet from an advertiser. After that, the scanner might respond with an CON_REQ packet, which is the first part of the connection establishment procedure. The packet lengths of packets involved in these procedures are taken from the values in ble_model_params_connection_establishment.h.

The process of advertising and scanning until there is the first reception is not accounted for in this function. This process is referred to as neighbor discovery and needs to be modeled separately.

<table>
<thead>
<tr>
<th>establishment-OrUpdate</th>
<th>1 =&gt; connection establishment procedure, 0 =&gt; connection parameter update</th>
</tr>
</thead>
<tbody>
<tr>
<td>scanType</td>
<td>Determines the connection comes about by continuous or periodic scanning. Only relevant for connection requests for initiators.</td>
</tr>
<tr>
<td>masterOrSlave</td>
<td>1 =&gt; master, 0 =&gt; slave</td>
</tr>
<tr>
<td>TcOld</td>
<td>Connection interval [s] before the parameter update. For connection establishment procedures, this value is ignored.</td>
</tr>
<tr>
<td>TcNew</td>
<td>The future connection interval [s] after the connection request or establishment procedure</td>
</tr>
</tbody>
</table>
Returns
Charge consumed by the parameter update

5.4 src/ble_model_discovery.h File Reference

Energy model for device discovery in BLE.

Data Structures

• struct __discoveryModelResult_t

Macros

• #define BLE_MODEL_DISCOVERY_FLOATING_EPSILON 1e-9

Typedefs

• typedef struct __discoveryModelResult_t discoveryModelResult_t

Functions

• double _ble_model_discovery_gausscdf (double x)
  Gets cumulative density function (CDF) of a Gaussian distribution. Code is taken from http://www.johndcook.com/cpp_phi.html It is released in public domain according to the website mentioned above: "This code is in the public domain. Do whatever you want with it, no strings attached."

• double _ble_model_discovery_normcdf (double x, double mu, double sigma)
  Calculates cumulative density function (CDF) of a (mu, sigma)-normal distribution by transforming the Gaussian CDF For the math behind this transformation, see http://en.wikipedia.org/wiki/Normal_distribution.

• double _ble_model_discovery_getApproxProbab (double mu, uint32_t n, double sigma, double t, double Tideal, double rhoMax)
  Computes the approximate probability of an advertising event having started before time t.

• discoveryModelResult_t _ble_model_discovery_getResultOnePhi (double epsilonHit, double Ta, double Ts, double ds, double phi, double rhoMax, double maxTime)
  Returns the model results (discovery-latency and discover-energy both for advertiser and scanner) for a given value offset phi.

• discoveryModelResult_t _ble_model_discovery_getResult (uint32_t nPoints, double epsilonHit, double Ta, double Ts, double ds, double rhoMax, double maxTime)
  Returns the model results (discovery-latency and discover-energy both for advertiser and scanner) for varying advertising offsets phi. nPoints different values of phi are examined.

5.4.1 Detailed Description

Energy model for device discovery in BLE. Call _ble_model_discovery_getResult() to get an estimate for the device-discovery latency and the corresponing energies spent by the master and the slave. This function requires high amounts of computational power. Set the parameters, especially nPoints, maxTime and epsilonHit carefully. You probably don’t want to use this function for online power management as it is computationally expensive.

Okt 2013, Philipp Kindt kindt@rcs.ei.tum.de
(c) 2013, Philipp Kindt (c) 2013, Lehrstuhl für Realzeit-Computersysteme (RCS), Technische Universität München (TUM)
This file is part of bleemod.

bleemod is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

bleemod is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with bleemod. If not, see http://www.gnu.org/licenses/

5.4.2 Typedef Documentation

5.4.2.1 typedef struct _discoveryModelResult_t discoveryModelResult_t

Result of the model for device discovery.

5.4.3 Function Documentation

5.4.3.1 double _ble_model_discovery_gausscdf ( double x)

Gets cumulative density function (CDF) of a Gaussian distribution. Code is taken from http://www.johndcook.com/cpp_phi.html It is released in public domain according to the website mentioned above: "This code is in the public domain. Do whatever you want with it, no strings attached."

Parameters

| x | value of the random variable. |

Returns

Value of Gaussian CDF for x

Examples:

examples.c.

5.4.3.2 double _ble_model_discovery_getApproxProbab ( double mu, uint32_t n, double sigma, double t, double TaIdeal, double rhoMax )

Computes the approximate probability of an advertising event having started before time t.

Parameters

| mu | Mean value of the starting time of the advertising event (=TaReal) |
| n | Number of the advertising event. 0 = > The event right after phi. 1 = > Event after one advertising interval, 2 = > Event after two advertising intervals |
| sigma | Standard deviation of the starting time of the advertising event |
| t | Time to evaluate the CDF |
| TaIdeal | Ideal point in time the advertising event starts(when no random advertising delay would exist) |
| rhoMax | Maximum random advDelay |

Returns

Approximate probability that the advertising event starts before t
5.4.3.3 discoveryModelResult_t ble_model_discovery_getResultOnePhi ( double epsilonHit, double Ta, double Ts, double ds, double phi, double rhoMax, double maxTime )

Returns the model results (discovery-latency and discover-energy both for advertiser and scanner) for a given value offset phi.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \epsilon_{\text{Hit}} )</td>
<td>The hit probability of all advertising events examined. As soon as the algorithm has reached this probability, it will consider the results as stable and stop. The closer this value becomes to one, the more precise the results become, but the algorithm takes longer as more advertising events are examined. Example: The first advertising event hits with probability 0.5, the second with 0.25 and the third with 0.15. The hit probability would be ( 0.5 + 0.25 + 0.15 = 0.9 ) If ( \epsilon_{\text{Hit}} ) is 0.89, the algorithm would end after three advertising events.</td>
</tr>
<tr>
<td>( \tau_a )</td>
<td>Advertising interval [s]</td>
</tr>
<tr>
<td>( T_s )</td>
<td>Scan interval [s]</td>
</tr>
<tr>
<td>( \delta_br )</td>
<td>Scan window [s]</td>
</tr>
<tr>
<td>( \phi )</td>
<td>Offset of the first scan event (n=0) from the beginning of the scanning process</td>
</tr>
<tr>
<td>( \rho_{\text{Max}} )</td>
<td>Maximum advertising delay [s]. Should be 10 ms according to the BLE specification</td>
</tr>
<tr>
<td>( \maxTime )</td>
<td>The maximum discovery latency possible. After that, the algorithm stops due to performance reasons</td>
</tr>
</tbody>
</table>

Returns

Discovery latency and the discovery energy spent by the advertiser and the scanner

5.4.3.4 double _ble_model_discovery_normcdf ( double \( x \), double \( \mu \), double \( \sigma \) )

Calculates cumulative density function (CDF) of a (\( \mu \), \( \sigma \))-normal distribution by transforming the Gaussian CDF For the math behind this transformation, see [http://en.wikipedia.org/wiki/Normal_distribution](http://en.wikipedia.org/wiki/Normal_distribution).

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>Value to evaluate</td>
</tr>
<tr>
<td>( \mu )</td>
<td>Mean value of the distribution</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>Standard deviation of the distribution</td>
</tr>
</tbody>
</table>

Returns

Value of the CDF evaluated at \( x \)

5.4.3.5 discoveryModelResult_t ble_model_discovery_getResult ( uint32_t \( n\text{Points} \), double \( \epsilon_{\text{Hit}} \), double \( \tau_a \), double \( T_s \), double \( \delta_br \), double \( \rho_{\text{Max}} \), double \( \maxTime \) )

Returns the model results (discovery-latency and discover-energy both for advertiser and scanner) for varying advertising offsets \( \phi \). \( n\text{Points} \) different values of \( \phi \) are examined.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n\text{Points} )</td>
<td>Number of advertising events ( \phi ) to be examined. The higher, the better the accuracy becomes but the longer the computation takes.</td>
</tr>
<tr>
<td>( \epsilon_{\text{Hit}} )</td>
<td>The hit probability of all advertising events examined for a particular ( \phi ). As soon as the algorithm has reached this probability, it will consider the results as stable and stop. The closer this value becomes to one, the more precise the results become, but the algorithm takes longer as more advertising events are examined. Example: The first advertising event hits with probability 0.5, the second with 0.25 and the third with 0.15. The hit probability would be ( 0.5 + 0.25 + 0.15 = 0.9 ) If ( \epsilon_{\text{Hit}} ) is 0.89, the algorithm would end after three advertising events</td>
</tr>
</tbody>
</table>
5.5 src/ble_model_params.h File Reference

<table>
<thead>
<tr>
<th>Ta</th>
<th>Advertising interval [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ts</td>
<td>Scan interval [s]</td>
</tr>
<tr>
<td>ds</td>
<td>Scan window [s]</td>
</tr>
<tr>
<td>$\rho_{\text{Max}}$</td>
<td>Maximum advertising delay [s]. Should be 10 ms according to the BLE specification</td>
</tr>
<tr>
<td>$maxTime$</td>
<td>The maximum discovery latency possible. After that, the algorithm stops due to performance reasons</td>
</tr>
</tbody>
</table>

Returns

Discovery latency and the discovery energy spent by the advertiser and the scanner

Examples:

examples.c.

5.5 src/ble_model_params.h File Reference

master include for all numerical values.

```
#include "ble_model_params_general.h"
#include "ble_model_params_connected.h"
#include "ble_model_params_scanning.h"
#include "ble_model_params_connection_establishment.h"
```

5.5.1 Detailed Description

master include for all numerical values. The only purpose of this file is including further headers that provide all numerical data values for the model parameters.

(c) 2013, Philipp Kindt (c) 2013, Lehrstuhl für Realzeit-Computersysteme (RCS), Technische Universität München (TUM)

This file is part of bleemod.

bleemod is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

bleemod is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with bleemod. If not, see http://www.gnu.org/licenses/

5.6 src/ble_model_params_connected.h File Reference

Device-dependant model params for Bluegiga BLE112 devices in connected mode.

Macros

- `#define BLE_MODEL_PARAMS_CONNECTED_H_`
- `#define BLE_E_MOD_C_DHEAD 0.578e-3`

Duration of head phase [s].

Generated on Tue Mar 11 2014 19:07:26 for Precise Energy Model for the Bluetooth Low Energy Standard by Doxygen
• #define BLE_E_MOD_C_DPRE 0.305e-3
  Duration of preprocessing phase [s].

• #define BLE_E_MOD_C_DCPRE 0.073e-3
  Duration of communication preamble phase [s].

• #define BLE_E_MOD_C_DPRERX 0.123e-3
  Duration of the prerx phase [s] for the master and for the slave except first rx-phase of slave within an event. The first duration of a slave is longer, see BLE_E_MOD_C_DPRERX_SL1. The prerx phase is the phase where the receiver is switched on, but no bits are transmitted. Therefore, the rx-phase is by dprerx longer than 8 mikroseconds * bytes received.

• #define BLE_E_MOD_C_DPRERX_SL1 0.388e-3
  Duration of the first prerx phase [s] of a slave. It is longer than different prerx phases and not related to window-widening. see BLE_E_MOD_C_DPRERX_SL.

• #define BLE_E_MOD_C_DRXTX 0.08e-3
  Duration of the Rx2Tx-phase [s].

• #define BLE_E_MOD_C_DPRETX 0.053e-3
  Duration of the pretx phase [s] (tx-phase is longer than 8 mikroseconds * bytes sent as the radio has to prepare)

• #define BLE_E_MOD_C_DTXRX 0.057e-3
  Duration of the Rx2Rx-phase [s].

• #define BLE_E_MOD_C_DTRA 0.066e-3
  Duration of the transient phase [s].

• #define BLE_E_MOD_C_DPOST 0.860e-3
  Duration of the postprocessing phase [s].

• #define BLE_E_MOD_C_DTAIL 0.08e-3
  Duration of the tail phase [s].

• #define BLE_E_MOD_C_IHEAD 5.924e-3
  Current magnitude of the head phase [A].

• #define BLE_E_MOD_C_IPRE 7.691e-3
  Current magnitude of the preprocessing phase [A].

• #define BLE_E_MOD_C_ICPRE 12.238e-3
  Current magnitude of the communication preamble phase [A].

• #define BLE_E_MOD_C_IRX 26.505e-3
  Current magnitude of the reception phase [A].

• #define BLE_E_MOD_C_IRXTX 14.128e-3
  Current magnitude of the Rx2Tx phase [A].

• #define BLE_E_MOD_C_ITX 36.445e-3
  Current magnitude of the Tx phase [A].

• #define BLE_E_MOD_C_ITXRX 15.125e-3
  Current magnitude of the Tx2Rx phase [A].

• #define BLE_E_MOD_C_ITRA 11.636e-3
  Current magnitude of the transient phase [A].

• #define BLE_E_MOD_C_IPOST 7.980e-3
  Current magnitude of the postprocessing phase [A].

• #define BLE_E_MOD_C_ITAIL 4.129e-3
  Current magnitude of the tail phase [A].

• #define BLE_E_MOD_C_QTO -1.2e-6
  Communication sequence correction offset [As].
5.6.1 Detailed Description

Device-dependant model params for Bluegiga BLE112 devices in connected mode. This file contains device-dependant values for the connected mode for Bluegiga BLE112-devices. The values have been optimized for the slave role, but should however match also master modes.

Change the values for different devices. For the model, please refer to the publication "A precise energy model for the Bluetooth Low Energy Protocol" by Philipp Kindt (c) 2013, Philipp Kindt (c) 2013, Lehrstuhl für Realzeit-Computersysteme (RCS), Technische Universität München (TUM)

This file is part of bleemod.

bleemod is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

bleemod is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with bleemod. If not, see http://www.gnu.org/licenses/

5.7 src/ble_model_params_connection_establishment.h File Reference

Energy model params for connection establishment and connection parameter updates.

Macros

• #define BLE_MODEL_PARAMS_CONNECTION_ESTABLISHMENT_H_
• #define BLE_E_MOD_CE_DTWO_CU 0
  Transmit window offset for connection update procedures.
• #define BLE_E_MOD_CE_DTWO_CR(TcNew) ((TcNew > 0.0125)?(TcNew - 0.006454):(0.389*TcNew+0.000484))
  Transmit window offset for connection request procedures. Contains a nonlinear formula depending on the future connection interval TcNew.
• #define BLE_E_MOD_CE_DTW(TcNew) 0.003
  Transmit window. Might depend on the future connection interval TcNew; for BLE112 devices it does not.
• #define BLE_E_MOD_CE_DP(TcNew) BLE_E_MOD_CE_DTW(TcNew)/2.0
  The average time the transmission begins after the beginning of the transmit window.
• #define BLE_E_MOD_CE_ADV_IND_PKG_LEN 37
  Number of bytes sent in an ADV_IND advertising packet by the advertiser.
• #define BLE_E_MOD_CE_CON_REQ_LEN 44
  Number of bytes sent in a CONNECT_REQ packet by the initiator (former scanner)
• #define BLE_E_MOD_CE_CON_UP_LEN 22
  Number of bytes sent in an LL_CONNECTION_UPDATE_REQ packet by the master.
• #define BLE_E_MOD_CE_CON_UP_SLRSP_LEN 10
  Number of bytes sent by the slave to the master in the event an LL_CONNECTION_UPDATE_REQ packet has been received.
• #define BLE_E_MOD_CE_CON_UP_TXPOWER 3
  Tx power level for connection update.
5.7.1 Detailed Description

Energy model params for connection establishment and connection parameter updates. 2.10.2013, Philipp Kindt
kindt@rcs.el.tum.de

This file contains parameter values for BLE112-devices for connection request procedures and connection parameter
updates.

(c) 2013, Philipp Kindt (c) 2013, Lehrstuhl für Realzeit-Computersysteme (RCS), Technische Universität München
(TUM)

This file is part of bleemod.

bleemod is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later
version.

bleemod is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public
License for more details.

You should have received a copy of the GNU Lesser General Public License along with bleemod. If not, see http://
www.gnu.org/licenses/

5.8 src/ble_model_params_general.h File Reference

Device-dependant model params for Bluegiga BLE112 devices that are independant from the mode (con-
nected/advertising/scanning/...)

Macros

- #define BLE_MODEL_PARAMS_GENERAL_H_
- #define BLE_E_MOD_G_SCA 50
- #define BLE_E_MOD_G_ISL 0.9e-6

5.8.1 Detailed Description

Device-dependant model params for Bluegiga BLE112 devices that are independant from the mode (con-
nected/advertising/scanning/...) This file contains device-dependant values for Bluegiga BLE112-devices that are
independant from the mode.

Change the values for different devices. For the model, please refer to the publication "A precise energy model for
the Bluetooth Low Energy Protocol" by Philipp Kindt

(c) 2013, Philipp Kindt (c) 2013, Lehrstuhl für Realzeit-Computersysteme (RCS), Technische Universität München
(TUM)

This file is part of bleemod.

bleemod is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later
version.

bleemod is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied
warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public
License for more details.

You should have received a copy of the GNU Lesser General Public License along with bleemod. If not, see http://
www.gnu.org/licenses/
Model parameters for scan events for BLE112-devices.

Macros

- `#define BLE_MODEL_PARAMS_SCANNING_H_`
- `#define BLE_E_MOD_SCAN_DPRE 0.700e-3`
  Duration [s] for wakeup & preprocessing for scan events.
- `#define BLE_E_MOD_SCAN_DRXTX 0.115e-3`
  Duration [s] for switching from the reception of the advertising packet to the sending of the SCAN_REQ / CON_REQ packet.
- `#define BLE_E_MOD_SCAN_DPRETX 0.014e-3`
  The TX phase of the scan request / connection request packet or whatever is sent is by this duration longer than 8 mikroseconds * bytes sent. Unit: [s].
- `#define BLE_E_MOD_SCAN_DTXRX 0.089e-3`
  Duration [s] for switching from sending the SCAN_REQ packet to receiving the SCAN_RESP packet. In the case of a CON_REQ packet, this phase does not occur.
- `#define BLE_E_MOD_SCAN_DPRERX 0.074e-3`
  The reception of the scan response is by this duration longer than 8 mikroseconds * bytes received. In the case of a CON_REQ packet, this phase does not occur. Unit: [s].
- `#define BLE_E_MOD_SCAN_DTXRX 0.377e-3`
  Duration [s] for switching from the reception of a SCAN_REQ packet to the continuation of the scanning. In the case of a CON_REQ packet, this phase does not occur.
- `#define BLE_E_MOD_SCAN_DPOST 0.816e-3`
  Duration [s] of the postprocessing phase of a scan event.
- `#define BLE_E_MOD_SCAN_DWOFFSET -1.85e-3`
  The scan window in the power curve usually is longer than the ideal scan window that was set by the application. Usually, this value is negative. Unit: [s].
- `#define BLE_E_MOD_SCAN_DCHCH 1.325e-3`
  Duration [s] for channel changing in continuous scanning.
- `#define BLE_E_MOD_SCAN_IPRE 7.087e-3`
  Current magnitude [A] of wake up & preprocessing phase.
- `#define BLE_E_MOD_SCAN_IRX 26.399e-3`
  Current magnitude [A] of Rx phase.
- `#define BLE_E_MOD_SCAN_IRXTX 15.011e-3`
  Current magnitude [A] of Rx2Tx phase.
- `#define BLE_E_MOD_SCAN_ITX 35.999e-3`
  Current magnitude [A] of Tx phase.
- `#define BLE_E_MOD_SCAN_ITXRX 16.670e-3`
  Current magnitude [A] of Tx2Rx phase.
- `#define BLE_E_MOD_SCAN_IRXS 26.426e-3`
  Current magnitude [A] of “Rx of scan response” phase.
- `#define BLE_E_MOD_SCAN_IRXRX 9.633e-3`
  Current magnitude [A] for channel changing in constant scanning.
- `#define BLE_E_MOD_SCAN_QCTX -0.2264e-6`
  Correction offset Tx [As].
- `#define BLE_E_MOD_SCAN_QCRX -0.1350e-6`
  Correction offset Rx [As].
5.9.1 Detailed Description

Model parameters for scan events for BLE112-devices. (c) 2013, Philipp Kindt (c) 2013, Lehrstuhl für Realzeit-Computersysteme (RCS), Technische Universität München (TUM)

This file is part of bleemod.

bleemod is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

bleemod is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with bleemod. If not, see http://www.gnu.org/licenses/

5.10 src/bble_model_scanning.h File Reference

Energy model for BLE scan events.

#include "ble_model_params_scanning.h"
#include "inttypes.h"

typedef enum
    _ble_e_model_sc_event_type_t ble_e_model_sc_event_type_t

The type of scan event.

typedef enum
    _ble_e_model_sc_scan_type_t ble_e_model_sc_scan_type

Determines weather we use continuous or periodic scanning.

typedef enum _ble_e_model_sc_event_type_t {
    SC_EVENT_TYPE_NO_RECEPTION, SC_EVENT_TYPE_PASSIVE_SCANNING, SC_EVENT_TYPE_ACTIVE_SCANNING, SC_EVENT_TYPE_CON_REQ,
    SC_EVENT_TYPE_CON_REQ_OFFSET, SC_EVENT_TYPE_ABORTED
}

The type of scan event.

typedef enum _ble_e_model_sc_scan_type_t { SC_SCAN_TYPE_PERIODIC, SC_SCAN_TYPE_CONTINUOUS }

Determines weather we use continuous or periodic scanning.

Functions

double ble_e_model_sc_getChargeScanEvent (double scanWindow, ble_e_model_sc_event_type_t eventType, ble_e_model_sc_scan_type scanType, uint8_t nBytesAdvInd, uint8_t nBytesTx, uint8_t nBytesRx, double receptionAfterTime)

Calculates the charge consumed by a scan event. This function calculates the charge consumed by a scan event. Different event types are supported. Further, it is distinguished between continuous scanning and periodic scanning. For periodic scanning, each event begins with preprocessing and ends with postprocessing. For continuous scanning, no beginning and end can be distinguished. Therefore, the event duration is determined by definition: The event begins when a scanner begins scanning on a certain channel and ends with the end of the channel-changing to the next channel. Therefore, each scan event contains on channel-changing phase.
5.10 Detailed Description

Energy model for BLE scan events. 2.10.2013, Philipp Kindt kindt@rcs.ei.tum.de (c) 2013, Philipp Kindt (c) 2013, Lehrstuhl für Realzeit-Computersysteme (RCS), Technische Universität München (TUM)

This file is part of bleemod.

bleemod is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

bleemod is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with bleemod. If not, see http://www.gnu.org/licenses/

5.10.2 Enumeration Type Documentation

5.10.2.1 enum _ble_e_model_sc_event_type_t

The type of scan event.

Enumerator

**SC_EVENT_TYPE_NO_RECEPTION**  Scan event that does not receive anything.

**SC_EVENT_TYPE_PASSIVE_SCANNING**  Scan event for passive scanning. No matter if advertising packets are received or not, this type is valid for passive scanning. Only exception: A CON_REQ packet is sent.

**SC_EVENT_TYPE_ACTIVE_SCANNING**  Scan event for active scanning without connecting. An advertising packet is received within this event, otherwise, it would be idle scanning and **SC_EVENT_TYPE_NO_RECEPTION** must be used.

**SC_EVENT_TYPE_CON_REQ**  Connection request packet is sent within that scan event. Takes into account whole scan event.

**SC_EVENT_TYPE_CON_REQ_OFFSET**  Connection request packet is sent within that scan event. This type only takes into account the energy spent for the additional energy compared to an idle scan event without reception which is aborted after the reception of an ADV_IND packet. This energy consists of the CON_REQ packet and drxtx, only! It is used for compatibility with the discovery energy model: To get the energy for discovery + connection, add the discovery energy and a packet of this type!

**SC_EVENT_TYPE_ABORTED**  Synonym for **SC_EVENT_TYPE_NO_RECEPTION**, but ends after reception-AfterTime of **ble_e_model_sc_getChargeScanEvent()**, leading to a shorter effective scan duration. It should be used for 'aborted' passive scanning event aborted after a certain amount of time due to a successful event. **SC_EVENT_TYPE_CON_REQ_OFFSET** + **SC_EVENT_TYPE_ABORTED** should model the whole event.

5.10.2.2 enum _ble_e_model_sc_scan_type_t

Determines weather we use continuous or periodic scanning.

Enumerator

**SC_SCAN_TYPE_PERIODIC**  Scan window < scan interval => periodic scanning.

**SC_SCAN_TYPE_CONTINUOUS**  Scan window = scan interval => continuous scanning.
5.10.3 Function Documentation

5.10.3.1 double ble_e_model_sc_getChargeScanEvent ( double scanWindow, ble_e_model_sc_event_type_t eventType, ble_e_model_sc_scan_type scanType, uint8_t nBytesAdvInd, uint8_t nBytesTx, uint8_t nBytesRx, double receptionAfterTime )

Calculates the charge consumed by a scan event. This function calculates the charge consumed by a scan event. Different event types are supported. Further, it is distinguished between continuous scanning and periodic scanning. For periodic scanning, each event begins with preprocessing and ends with postprocessing. For continuous scanning, no beginning and end can be distinguished. Therefore, the event duration is determined by definition: The event begins when a scanner begins scanning on a certain channel and ends with the end of the channel-changing to the next channel. Therefore, each scan event contains one channel-changing phase.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scanWindow</td>
<td>Scan window [s]. For SC_EVENT_TYPE_CON_REQ_OFFSET, this parameter is discarded</td>
</tr>
<tr>
<td>eventType</td>
<td>Type of the scan event that occurs.</td>
</tr>
<tr>
<td>scanType</td>
<td>Determines whether periodic scanning (scan window &lt; scan interval) or continuous scanning (scan window = scan interval) takes place.</td>
</tr>
<tr>
<td>nBytesAdvInd</td>
<td>Bytes of the ADV_IND packet received. This value is currently ignored, it may be set to any value. It is reserved for future use.</td>
</tr>
<tr>
<td>nBytesTx</td>
<td>Number of bytes sent in a scan request or connection request packet by the master. Only used for SC_EVENT_TYPE_ACTIVE_SCANNING, SC_EVENT_TYPE_CON_REQ and SC_EVENT_TYPE_CON_REQ_OFFSET</td>
</tr>
<tr>
<td>nBytesRx</td>
<td>Number of bytes received in a scan response. Only used for SC_EVENT_TYPE_ACTIVE_SCANNING.</td>
</tr>
<tr>
<td>receptionAfterTime</td>
<td>Number of seconds beginning from the scan event after which an advertising packet has been received completely. As this value is unknown most times, the beginning of the reception can be inserted</td>
</tr>
</tbody>
</table>

Returns

Charge consumed by the scan event [As]

Examples:

tables/c.
#include "../src/ble_model.h"
#include <stdio.h>

int main() {
    double dc, dseq, dwhole;

    //get the duration of all constant parts of a connection event. (Preprocessing, Postprocessing,...)
    dc = ble_e_model_c_getDurationConstantParts();

    //get the duration of the connection sequence (all non-constant parts of the model)
    dseq = ble_e_model_c_getDurationSequencesSamePayload(1,0.1,5,10,22,3);

    //get the duration of the whole event, this should be the sum of dc and dseq
    dwhole = ble_e_model_c_getDurationEventSamePayload(1,0.1,5,10,22,3);

    //now do the same with the charge of these phases
    double chargeC, chargeSeq, chargeWhole;
    chargeC = ble_e_model_c_getChargeConstantParts();
    chargeSeq = ble_e_model_c_getChargeSequencesSamePayload(1,0.1,5,10,22,3);
    chargeWhole = ble_e_model_c_getChargeEventSamePayload(1,0.1,5,10,22,3);

    printf("Duration: const %.15f - seq: %.15f, whole: %.15f\n", dc, dseq, dwhole);
    printf("Charge: const %.15f - seq: %.15f, whole: %.15f\n", chargeC, chargeSeq, chargeWhole);

    //now we’re not interested in the events only, but also in the sleeping phase of a connection event
    double chargeInterval;
    chargeInterval = ble_e_model_c_getChargeConnectionIntervalSamePayload(0,0.1,5,10,22,3);
    printf("Charge for connection interval: %.15f\n", chargeInterval);

    //now get the charge for a master establishing a connection. We are only interested in the
    //additional energy that is not accounted for in the energy model for device discovery as described in the paper, and
    //not the whole model.
    //Therefore, we use \ref SC_EVENT_TYPE_CON_REQ_OFFSET
    double croCharge = ble_e_model_sc_getChargeScanEvent(0.025, SC_EVENT_TYPE_CON_REQ_OFFSET, SC_SCAN_TYPE_PERIODIC, BLE_E_MOD_CE_ADV_IND_PKG_LEN, BLE_E_MOD_CE_CON_REQ_LEN, 0.0, 0.0, 0.0125);
    printf("Connection Request offset charge: %.15f\n", croCharge);

    //now we calculate the energy consumption of a scan event with active scanning that receives and
    //replies to an advertising event
    double croActScan = ble_e_model_sc_getChargeScanEvent(0.025, SC_EVENT_TYPE_ACTIVE_SCANNING, SC_SCAN_TYPE_PERIODIC, 28, 22, 34, 0.125);
    printf("Active Scanning Event: %.15f\n", croActScan);

    _ble_model_discovery_gausscdf(33.0);

    //Calculate the latency and energy consumption of device discovery for Ta = 2.55s, Ts = 2.56s and
    //ds = 64ms.
    discoveryModelResult_t result = ble_model_discovery_getResult(100,0.9999,2.55,2.56,0.064,0.01,1000);
    printf("discovery latency: %.15f\n", result.discoveryLatency);
    printf("discovery energy: advertiser: %.15f, scanner: %.15f\n", result.chargeAdv, result.chargeScan);

    return 0;
}
ble_e_model_sc_event_type_t
ble_model_scanning.h, 29
ble_e_model_sc_scan_type_t
ble_model_scanning.h, 29
ble_model_discovery_gausscdf
ble_model_discovery.h, 20
ble_model_discovery_getApproxProbab
ble_model_discovery.h, 20
ble_model_discovery_getResultOnePhi
ble_model_discovery.h, 20
discoveryModelResult_t, 7
discoveryModelResult_t
ble_model_scanning.h
SC_EVENT_TYPE_ABORTED, 29
SC_EVENT_TYPE_ACTIVE_SCANNING, 29
SC_EVENT_TYPE_CON_REQ, 29
SC_EVENT_TYPE_CON_REQ_OFFSET, 29
SC_EVENT_TYPE_NO_RECEPTION, 29
SC_EVENT_TYPE_PASSIVE_SCANNING, 29
SC_SCAN_TYPE_CONTINUOUS, 29
SC_SCAN_TYPE_PERIODIC, 29
ble_e_model_c_getChargeConnectionInterval
ble_model_connected.h, 11
ble_e_model_c_getChargeConnectionIntervalSamePayload
ble_model_connected.h, 12
ble_e_model_c_getChargeConstantParts
ble_model_connected.h, 12
ble_e_model_c_getChargeEvent
ble_model_connected.h, 12
ble_e_model_c_getChargeEventSamePayload
ble_model_connected.h, 13
ble_e_model_c_getChargeSequences
ble_model_connected.h, 13
ble_e_model_c_getChargeSequencesSamePayload
ble_model_connected.h, 14
ble_e_model_c_getDurationConstantParts
ble_model_connected.h, 14
ble_e_model_c_getDurationEvent
ble_model_connected.h, 14
ble_e_model_c_getDurationEventSamePayload
ble_model_connected.h, 14
ble_e_model_c_getDurationEventSamePayload
ble_model_connected.h, 16
ble_e_model_c_getDurationSequences
ble_model_connected.h, 16
ble_e_model_c_getDurationSequencesSamePayload
ble_model_connected.h, 17
ble_e_model_ce_getChargeForConnectionProcedure
ble_model_connection_establishment.h, 18
ble_e_model_sc_getChargeScanEvent
ble_model_scanning.h, 30
ble_model_connected.h
ble_e_model_c_getChargeConnectionInterval, 11
ble_e_model_c_getChargeConnectionIntervalSamePayload, 12
ble_e_model_c_getChargeConstantParts, 12
ble_e_model_c_getChargeEvent, 12
ble_e_model_c_getChargeEventSamePayload, 13
ble_e_model_c_getChargeSequencesSamePayload, 14
discoveryModelResult_t, 20
discoveryModelResult_t
discoveryModelResult_t
ble_model_discovery.h
ble_model_discovery_gausscdf, 20
ble_model_discovery_getApproxProbab, 20
ble_model_discovery_getResultOnePhi, 20
ble_model_discovery_normcdf, 20
ble_model_discovery_getResult, 22
discoveryModelResult_t
ble_model_discovery.h, 22
ble_model_discovery.h
ble_e_model_sc_event_type_t, 29
ble_e_model_sc_scan_type_t, 29
ble_model_scanning.h
SC_EVENT_TYPE_ABORTED, 29
SC_EVENT_TYPE_ACTIVE_SCANNING, 29
SC_EVENT_TYPE_CON_REQ, 29
SC_EVENT_TYPE_CON_REQ_OFFSET, 29
SC_EVENT_TYPE_NO_RECEPTION, 29
SC_EVENT_TYPE_PASSIVE_SCANNING, 29
SC_SCAN_TYPE_CONTINUOUS, 29
ble_model_scanning.h
ble_e_model_sc_...
SC_SCAN_TYPE_PERIODIC
   ble_model_scanning.h, 29
src/ble_model.h, 9
src/ble_model_connected.h, 10
src/ble_model_connection_establishment.h, 17
src/ble_model_discovery.h, 19
src/ble_model_params.h, 23
src/ble_model_params_connected.h, 23
src/ble_model_params_connection_establishment.h, 25
src/ble_model_params_general.h, 26
src/ble_model_params_scanning.h, 27
src/ble_model_scanning.h, 28