User's Manual

Bluetooth® Low Energy Protocol Stack

API Reference Manual: CPP

Renesas MCU

Target Device

RL78/G1D

Renesas Electronics
www.renesas.com

Rev.1.02 Apr 2015

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(2012.4)

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

How to Use This Manual

1. Purpose and Target Readers

This manual describes the API (Application Program Interface) of the Cycling Power profile (CPP) of the Bluetooth Low Energy protocol stack (BLE software), which is used to develop Bluetooth applications that incorporate the Renesas Bluetooth low energy microcontroller RL78/G1D. It is intended for users designing application systems incorporating this software. A basic knowledge of microcontrollers and Bluetooth low energy is necessary in order to use this manual.

Related documents

The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

Document Name	Document No.
Bluetooth Low Energy Protocol Stack	
User's Manual	R01UW0095E
API Reference Manual: Basics	R01UW0088E
API Reference Manual: FMP	R01UW0089E
API Reference Manual: PXP	R01UW0090E
API Reference Manual: HTP	R01UW0091E
API Reference Manual: BLP	R01UW0092E
API Reference Manual: HOGP	R01UW0093E
API Reference Manual: ScPP	R01UW0094E
API Reference Manual: HRP	R01UW0097E
API Reference Manual: CSCP	R01UW0098E
API Reference Manual: CPP	This manual
API Reference Manual: GLP	R01UW0103E
API Reference Manual: TIP	R01UW0106E
API Reference Manual: RSCP	R01UW0107E
API Reference Manual: ANP	R01UW0108E
API Reference Manual: PASP	R01UW0109E
API Reference Manual: LNP	R01UW0113E
Application Note: Sample Program	R01AN1375E
Application Note: rBLE Command Specification	R01AN1376E

List of Abbreviations and Acronyms

Abbreviation	Full Form	Remark
ANP	Alert Notification Profile	
ANS	Alert Notification Service	
API	Application Programming Interface	
ATT	Attribute Protocol	
BAS	Battery Service	
BB	Base Band	
BD_ADDR	Bluetooth Device Address	
BLE	Bluetooth low energy	
BLP	Blood Pressure Profile	
BLS	Blood Pressure Service	
CPP	Cycling Power Profile	
CPS	Cycling Power Service	
CSCP	Cycling Speed and Cadence Profile	
CSCS	Cycling Speed and Cadence Service	
CSRK	Connection Signature Resolving Key	
CTS	Current Time Service	
DIS	Device Information Service	
EDIV	Encrypted Diversifier	
FMP	Find Me Profile	
GAP	Generic Access Profile	
GATT	Generic Attribute Profile	
GLP	Glucose Profile	
GLS	Glucose Service	
HCI	Host Controller Interface	
HID	Human Interface Device	
HIDS	HID Service	
HOGP	HID over GATT Profile	
HRP	Heart Rate Profile	
HRS	Heart Rate Service	
HTP	Health Thermometer Profile	
HTS	Health Thermometer Service	
IAS	Immediate Alert Service	
IRK	Identity Resolving Key	
L2CAP	Logical Link Control and Adaptation Protocol	
LE	Low Energy	

Abbreviation	Full Form	Remark
LL	Link Layer	
LLS	Link Loss Service	
LNP	Location and Navigation Profile	
LNS	Location and Navigation Service	
LTK	Long Term Key	
MCU	Micro Controller Unit	
MITM	Man-in-the-middle	
MTU	Maximum Transmission Unit	
NDCS	Next DST Change Service	
OOB	Out of Band	
OS	Operating System	
PASP	Phone Alert Status Profile	
PASS	Phone Alert Status Service	
PXP	Proximity Profile	
RF	Radio Frequency	
RSCP	Running Speed and Cadence Profile	
RSCS	Running Speed and Cadence Service	
RSSI	Received Signal Strength Indication	
RTUS	Reference Time Update Service	
ScPP	Scan Parameters Profile	
ScPS	Scan Parameters Service	
SM	Security Manager	
SMP	Security Manager Protocol	
STK	Short Term Key	
TIP	Time Profile	
TK	Temporary Key	
TPS	Tx Power Service	
UART	Universal Asynchronous Receiver Transmitter	
UUID	Universal Unique Identifier	

Abbreviation	Full Form	Remark
APP	Application	
CSI	Clocked Serial Interface	
IIC	Inter-Integrated Circuit	
RSCIP	Renesas Serial Communication Interface Protocol	
VS	Vendor Specific	

Contents

1. Overview

This manual describes the API (Application Program Interface) of the Cycling Power profile (CPP) of the Bluetooth Low Energy protocol stack (BLE software), which is used to develop Bluetooth applications that incorporate Renesas Bluetooth low energy microcontroller RL78/G1D.

For details about the organization and features of BLE software, see the Bluetooth Low Energy Protocol Stack User's Manual.

2. Common Definitions

This section describes the definitions common to the API of each profile.

2.1 Service Definitions

This section describes the common definitions of services used by the API of multiple profiles.

- Declaration of enumerated type for alert level

```
enum RBLE_SVC_ALT_LVL_enum {
   RBLE_SVC_ALERT_NONE = 0x00,            No alert
   RBLE_SVC_ALERT_MILD,                   Mild alert
   RBLE_SVC_ALERT_HIGH                    High alert
};
```

- Declaration of enumerated type for PnP ID characteristic vendor ID field

```
enum RBLE_SVC_PNP_VENDOR_ID_enum {
   RBLE_SVC_SIG_ASSIGNED_ID = 0x01,       Vendor ID assigned by Bluetooth SIG
   RBLE_SVC_USB_ASSIGNED_ID               Vendor ID assigned by USB Implementer's
                                          Forum
};
```

- Declaration of enumerated type for Name Space field of Characteristic Presentation Format descriptor

```
enum RBLE_SVC_PRESEN_NAMESPASE_enum {
   RBLE_SVC_NAMESPACE_SIG = 0x01,         Defined by Bluetooth SIG
};
```

- Declaration of enumerated type for security level of Service

```
enum RBLE_SVC_SEC_LVL_enum {
   RBLE_SVC_SEC_NONE = 0x01,              No security
   RBLE_SVC_SEC_UNAUTH = 0x02,            Require unauthenticated pairing
   RBLE_SVC_SEC_AUTH = 0x04,              Require authenticated pairing
   RBLE_SVC_SEC_AUTZ = 0x08,              Require authorization
   RBLE_SVC_SEC_ENC = 0x10                Require encryption
};
```

- Declaration of enumerated type for connection types

```
enum RBLE_PRF_CON_enum {
    RBLE_PRF_CON_DISCOVERY = 0x00,        Configuration connection performed
                                          when connecting for the first time
    RBLE_PRF_CON_NORMAL                   Normal connection performed when
                                          connecting for the second and
                                          subsequent times
};
```

- Declaration of enumerated type for client configuration characteristic value

```
enum RBLE_PRF_CLIENT_CONFIG_enum {
    RBLE_PRF_STOP_NTFIND = 0x00,            Stop notification or indication of
                                            characteristic value.
    RBLE_PRF_START_NTF,                     Start notification of
                                            characteristic value.
    RBLE_PRF_START_IND                      Start indication of
                                            characteristic value.
};
```

- Declaration of enumerated type for server configuration characteristic value

```
enum RBLE_PRF_SERVER_CONFIG_enum {
    RBLE_PRF_STOP_BRD = 0x00,               Stop broadcast of characteristic value.
    RBLE_PRF_START_BRD                      Start broadcast of characteristic value.
};
```

2.2 Status Definitions

This section describes the status definitions used by the API of each profile.

- Declaration of enumerated type for rBLE status

```
enum RBLE_STATUS_enum {
   RBLE_OK = 0x00,                          Normal operation
   RBLE_PRF_ERR_INVALID_PARAM = 0x90,       Invalid parameter specified for
                                            setting or acquiring a characteristic
                                            value
   RBLE_PRF_ERR_INEXISTENT_HDL,             Invalid handle specified for setting
                                            or acquiring a characteristic value
   RBLE_PRF_ERR_STOP_DISC_CHAR_MISSING,     The characteristic value is missing.
   RBLE_PRF_ERR_MULTIPLE_IAS,               Multiple IASs exist.
   RBLE_PRF_ERR_INCORRECT_PROP,             Incorrect property
   RBLE_PRF_ERR_MULTIPLE_CHAR,              Multiple characteristic values exist.
   RBLE_PRF_ERR_NOT_WRITABLE,               Writing is not permitted.
   RBLE_PRF_ERR_NOT_READABLE,               Reading is not permitted.
   RBLE_PRF_ERR_REQ_DISALLOWED,             Requesting is not permitted.
   RBLE_PRF_ERR_NTF_DISABLED,               Notification is disabled.
   RBLE_PRF_ERR_IND_DISABLED,               Indication is disabled.
   RBLE_PRF_ERR_ATT_NOT_SUPPORTED,          The characteristic value is not
                                            supported.
};
```

Note: Statuses other than the above are described in *API Reference Manual: Basics*.

3. Cycling Power Profile

This section describes the API of the Cycling Power profile. The Cycling Power profile is used to enable a data collection device to obtain data from cycling sensors.

3.1 Definitions

This section describes the definitions used by the API of the Cycling Power profile.

- Declaration of macro definition for maximum number of MAGNITUDE information

```
#define RBLE_CPP_MAGNITUDE_MAX            9
```

- Declaration of macro definition for supported maximum number of sensor location information

```
#define RBLE_CPP_SENSORE_LOCATION_MAX     17
```

- Declaration of enumerated type for CPP event types

```
enum RBLE_CPP_EVENT_TYPE_enum {
    RBLE_CPP_EVENT_SENSOR_ENABLE_COMP = 0x01,    Sensor enable completion event
                                                 (Parameter: sensor_enable)
    RBLE_CPP_EVENT_SENSOR_DISABLE_COMP,          Sensor disable completion event
                                                 (Parameter: sensor_disable)
    RBLE_CPP_EVENT_SENSOR_ERROR_IND,             Sensor error indication event
                                                 (Parameter: error_ind)
    RBLE_CPP_EVENT_SENSOR_SEND_MEASUREMENTS_COMP,
                                                 Measurements send completion event
                                                 (Parameter: send_measurements)
    RBLE_CPP_EVENT_SENSOR_SEND_VECTOR_COMP,      Power vector measurement information
                                                 send completion event
                                                 (Parameter: send_vector)
    RBLE_CPP_EVENT_SENSOR_SEND_CP_CP_COMP,       CP control point send completion event
                                                 (Parameter: send_cp_cp)
    RBLE_CPP_EVENT_SENSOR_SEND_BATTERY_LEVEL_COMP,
                                                 Battery level send completion event
                                                 (Parameter: send_battery_level)
    RBLE_CPP_EVENT_SENSOR_CHG_CP_CP_IND,         CP control point change indication event
                                                 (Parameter: chg_cp_cp_ind)
    RBLE_CPP_EVENT_SENSOR_CFG_INDNTFBRD_IND,     Characteristic configuration change
                                                 indication event
                                                 (Parameter: cfg_indntfbrd_ind)
    RBLE_CPP_EVENT_SENSOR_COMMAND_DISALLOWED_IND,
                                                 Command disallowed indication event
                                                 (Parameter: cmd_disallowed_ind)
    RBLE_CPP_EVENT_COLLECTOR_ENABLE_COMP = 0x81,
                                                 Collector enable completion event
                                                 (Parameter: collector_enable)
    RBLE_CPP_EVENT_COLLECTOR_DISABLE_COMP,       Collector disable completion event
                                                 (Parameter: collector_disable)
```

```
    RBLE_CPP_EVENT_COLLECTOR_ERROR_IND,          Collector error indication event
                                                 (Parameter: error_ind)
    RBLE_CPP_EVENT_COLLECTOR_MEASUREMENTS_NTF,   Cycling power measurement notification
                                                 event
                                                 (Parameter: measurements_ntf)
    RBLE_CPP_EVENT_COLLECTOR_VECTOR_NTF,         Cycling power vector notification event
                                                 (Parameter: vector_ntf)
    RBLE_CPP_EVENT_COLLECTOR_CP_CP_IND,          CP control point indication event
                                                 (Parameter: cp_cp_ind)
    RBLE_CPP_EVENT_COLLECTOR_BATTERY_LEVEL_NTF,  Battery level notification event
                                                 (Parameter: battery_level_ntf)
    RBLE_CPP_EVENT_COLLECTOR_READ_CHAR_RESPONSE,
                                                 Characteristic value read request
                                                 response event
                                                 (Parameter: rd_char_resp)
    RBLE_CPP_EVENT_COLLECTOR_WRITE_CHAR_RESPONSE,
                                                 Characteristic value write request
                                                 response event
                                                 (Parameter: wr_char_resp)
    RBLE_CPP_EVENT_COLLECTOR_COMMAND_DISALLOWED_IND
                                                 Command disallowed indication event
                                                 (Parameter: cmd_disallowed_ind)
};
```

- Declaration of data type for CPP event types

```
typedef uint8_t RBLE_CPP_EVENT_TYPE;
```

- Declaration of data type for CPP Sensor event callback function

```
typedef void ( *RBLE_CPPS_EVENT_HANDLER )( RBLE_CPPS_EVENT *event );
```

- Declaration of data type for CPP Collector event callback function

```
typedef void ( *RBLE_CPPC_EVENT_HANDLER )( RBLE_CPPC_EVENT *event );
```

- Declaration of enumerated type for cycling power service/device information service characteristic codes

```
enum RBLE_CPPC_RD_CHAR_CODE_enum {
    RBLE_CPPC_RD_CPS_CM_CFG = 0x00,              Cycling power measurement notification
    RBLE_CPPC_RD_CPS_CM_BRD_CFG,                 Cycling power measurement broadcast
    RBLE_CPPC_RD_CPS_CV_CFG,                     Cycling power vector notification
    RBLE_CPPC_RD_CPS_CPCP_CFG,                   CP control point indication
    RBLE_CPPC_RD_CPS_CP_FEATURE,                 Cycling power feature
    RBLE_CPPC_RD_CPS_SL,                         Sensor location
    RBLE_CPPC_RD_DIS_MANUF,                      Sensor manufacturer name
    RBLE_CPPC_RD_DIS_MODEL,                      Sensor model number
    RBLE_CPPC_RD_DIS_SERNB,                      Sensor serial number
    RBLE_CPPC_RD_DIS_HWREV,                      Sensor hardware revision
    RBLE_CPPC_RD_DIS_FWREV,                      Sensor firmware revision
    RBLE_CPPC_RD_DIS_SWREV,                      Sensor software revision
```

```
    RBLE_CPPC_RD_DIS_SYSID,                     Sensor system ID
    RBLE_CPPC_RD_DIS_IEEE,                      Sensor IEEE certification
                                                information
    RBLE_CPPC_RD_BAS_BL,                        Sensor battery level information
    RBLE_CPPC_RD_BAS_BL_CFG,                    Sensor battery level notification
};
```

- Declaration of enumerated type for cycling power service and battery service characteristic value settings

```
enum RBLE_CPPC_WR_CHAR_CODE_enum {
    RBLE_CPPC_CYCPWR_MEAS_CODE = 0x01,          Cycling power measurement characteristic
                                                notification setting
    RBLE_CPPC_CYCPWR_MEAS_BRD_CODE,             Cycling power measurement characteristic
                                                broadcast setting
    RBLE_CPPC_CYCPWR_VCTR_CODE,                 Cycling power vector characteristic
                                                notification setting
    RBLE_CPPC_CYCPWR_CONTROL_POINT_CODE,        CP control point characteristic
                                                indication setting
    RBLE_CPPC_BATTERY_LEVEL_CODE                Battery level characteristic
                                                notification setting
};
```

- Declaration of enumerated type for sensor location characteristic setting

```
enum RBLE_CPPC_SENSOR_LOCATION_enum {
    RBLE_CPPC_SENSOR_OTHER = 0x00,              Other
    RBLE_CPPC_SENSOR_TOP_OF_SHOE,               Top of shoe
    RBLE_CPPC_SENSOR_IN_SHOE,                   in shoe
    RBLE_CPPC_SENSOR_HIP,                       Hip
    RBLE_CPPC_SENSOR_FRONT_WHEEL,               Front wheel
    RBLE_CPPC_SENSOR_LEFT_CRANK,                Left crank
    RBLE_CPPC_SENSOR_RIGHT_CRANK,               Right crank
    RBLE_CPPC_SENSOR_LEFT_PEDAL,                Left pedal
    RBLE_CPPC_SENSOR_RIGHT_PEDAL,               Right pedal
    RBLE_CPPC_SENSOR_FRONT_HUB,                 Front hub
    RBLE_CPPC_SENSOR_REAR_DROPOUT,              Rear dropout
    RBLE_CPPC_SENSOR_CHAINSTAY,                 Chainstay
    RBLE_CPPC_SENSOR_REAR_WHEEL,                Rear wheel
    RBLE_CPPC_SENSOR_REAR_HUB,                  Rear hub
    RBLE_CPPC_SENSOR_CHEST                      Chest
};
```

- Declaration of enumerated type for CP control point characteristic operation code setting

```
enum RBLE_CPP_CP_OP_CODE_enum {
    RBLE_CPP_OP_SET_CUMULATIVE_CODE = 0x01,     Set cumulative value
    RBLE_CPP_OP_UPDATE_SL_CODE,                 Update sensor location
    RBLE_CPP_OP_REQ_SUPPORTED_SL_CODE,          Request supported sensor locations
    RBLE_CPP_OP_SET_CRANK_LEN_CODE,             Set crank length
    RBLE_CPP_OP_REQ_CRANK_LEN_CODE,             Request crank length
```

```
    RBLE_CPP_OP_SET_CHAIN_LEN_CODE,              Set chain length
    RBLE_CPP_OP_REQ_CHAIN_LEN_CODE,              Request chain length
    RBLE_CPP_OP_SET_CHAIN_WEI_CODE,              Set chain weight
    RBLE_CPP_OP_REQ_CHAIN_WEI_CODE,              Request chain weight
    RBLE_CPP_OP_SET_SPAN_LEN_CODE,               Set span length
    RBLE_CPP_OP_REQ_SPAN_LEN_CODE,               Request span length
    RBLE_CPP_OP_START_OFFSET_COMPENSATION_CODE,  Start offset compensation
    RBLE_CPP_OP_MASK_CP_MEAS_CONTENT_CODE,       Mask cycling power measurement
                                                 characteristic content
    RBLE_CPP_OP_REQ_SAMPL_RATE_CODE,             Request sampling rate
    RBLE_CPP_OP_REQ_FACTORY_CALIB_DATE_CODE,     Request factory calibration date
    RBLE_CPP_OP_RESPONSE_CODE = 0x20             Response code
};
```

- Declaration of enumerated type for CP control point characteristic response setting

```
enum RBLE_CPP_CPCP_RES_CODE_enum {
    RBLE_CPP_RES_SUCCESS_CODE = 0x01,            Success
    RBLE_CPP_RES_NOT_SUPPORTED_CODE,             Op Code not supported
    RBLE_CPP_RES_INVALID_PARAM_CODE,             Invalid parameter
    RBLE_CPP_RES_OP_FAILED_CODE                  Operation failed
};
```

- Cycling power service characteristic information structures

```
typedef struct RBLE_CPP_SENSOR_PARAM_t {
    uint16_t        cp_meas_ntf_en;              Cycling power measurement
                                                 notification configuration value
    uint16_t        cp_meas_brd_en;              Cycling power measurement
                                                 broadcast configuration value
    uint16_t        cp_vector_ntf_en;            Cycling power vector
                                                 notification configuration value
    uint16_t        cp_cp_ind_en;                CP control point
                                                 indication configuration value
    uint16_t        battery_level_ntf_en;        Battery level
                                                 notification configuration value
    uint8_t         sensor_location;             Sensor location
    uint8_t         reserved;                    Reserved
}RBLE_CPP_SENSOR_PARAM;
```

- Cycling power measurement information structures

```
typedef struct RBLE_CPP_MEASUREMENTS_INFO_t{
    uint16_t        flags;                       Data field flag
    int16_t         instant_power;               Instantaneous power [w]
    uint8_t         pedal_balance;               Pedal power balance [%]
    uint8_t         reserved;                    Reserved
    uint16_t        accumulated_torque;          Accumulated torque [1/32 Nm]
    uint32_t        wheel_revolutions;           Cumulative wheel revolutions
    uint16_t        wheel_event;                 Last wheel event time [1/2048 second]
```

```
    uint16_t        crank_revolutions;          Cumulative crank revolutions
    uint16_t        crank_event;                Last crank event time [1/1024 second]
    uint16_t        max_force_magnitude;        Maximum force magnitude [N]
    uint16_t        min_force_magnitude;        Minimum force magnitude [N]
    uint16_t        max_torque_magnitude;       Maximum torque magnitude [1/32 Nm]
    uint16_t        min_torque_magnitude;       Minimum torque magnitude [1/32 Nm]
    uint16_t        max_angle;                  Maximum angle [degree]
    uint16_t        min_angle;                  Minimum angle [degree]
    uint16_t        top_dead_spot;              Top dead spot angle [degree]
    uint16_t        bottom_dead_spot;           Bottom dead spot angle [degree]
    uint16_t        accumulated_energy;         Accumulated energy [kJ]
}RBLE_CPP_TMEASUREMENTS_INFO;
```

- Cycling power vector information structure

```
typedef struct RBLE_CPP_VECTOR_INFO_t{
    uint8_t         flags;                      Data field flag
    uint8_t         reserved1;                  Reserved
    uint16_t        crank_revolutions;          Cumulative crank revolutions
    uint16_t        crank_event_time;           Last crank event time [1/1024 second]
    uint16_t        first_crank_angle;          First crank measurement angle [degree]
    uint8_t         array_num;                  Number of magnitude information
    uint8_t         reserved2;                  Reserved
    int16_t         magnitude[RBLE_CPP_MAGNITUDE_MAX];
                                                Instantaneous force magnitude array [N]
                                                or Instantaneous torque magnitude array
                                                [1/32 Nm]
} RBLE_CPP_VECTOR_INFO;
```

- CP control point setting information structure

```
typedef struct RBLE_CPP_WR_CONTROL_POINT_INFO_t{
    uint8_t         OpCode;                     Op Code
    uint8_t         reserved1;                  Reserved
    uint32_t        cumulative_value;           Cumulative value
    uint8_t         sensor_location;            Sensor location value
    uint8_t         reserved2;                  Reserved
    uint16_t        crank_length;               Crank length value [1/2mm]
    uint16_t        chain_length;               Chain length value [mm]
    uint16_t        chain_weight;               Chain weight value [g]
    uint16_t        span_length;                Span length value [mm]
    uint16_t        mask_meas_content;          Content mask
} RBLE_CPP_WR_CONTROL_POINT_INFO;
```

- Date and time information structures

```
typedef struct RBLE_DATE_TIME_t{
    uint16_t        year;                       Year
    uint8_t         month;                      Month
    uint8_t         day;                        Day
```

```
    uint8_t         hour;                       Hour
    uint8_t         min;                        Minute
    uint8_t         sec;                        Second
    uint8_t         reserved;                   Reserved
}RBLE_DATE_TIME;
```

- CP control point indication information structure

```
typedef struct RBLE_CPP_IND_CONTROL_POINT_INFO_t{
    uint8_t         OpCode;                     Op Code
    uint8_t         request_op_code;            Request Op Code
    uint8_t         response_value;             Response value
    uint8_t         reserved1;                  Reserved
    uint16_t        crank_length;               Crank length value [1/2mm]
    uint16_t        chain_length;               Chain length value [mm]
    uint16_t        chain_weight;               Chain weight value [g]
    uint16_t        span_length;                Span length value [mm]
    uint16_t        offset_compensation;        Offset compensation value
    uint16_t        sampling_rate;              Sampling rate
    uint16_t        reserved2;                  Reserved
    RBLE_DATE_TIME  stamp;                      Time stamp
} RBLE_CPP_IND_CONTROL_POINT_INFO;
```

- Cycling power service content structures

```
typedef struct RBLE_CPS_CONTENT_t{
    uint16_t        shdl;                       Cycling power service start handle
    uint16_t        ehdl;                       Cycling power service end handle
    uint16_t        cp_meas_char_hdl;           Cycling power measurement
                                                characteristic handle
    uint16_t        cp_meas_val_hdl;            Cycling power measurement
                                                characteristic value handle
    uint16_t        cp_meas_cfg_hdl;            Cycling power measurement client
                                                characteristic configuration
                                                descriptor handle
    uint16_t        cp_meas_brd_cfg_hdl;        Cycling power measurement server
                                                characteristic configuration
                                                descriptor handle
    uint8_t         cp_meas_prop;               Cycling power measurement
                                                characteristic property
    uint8_t         reserved1;                  Reserved
    uint16_t        cp_feature_char_hdl;        Cycling power feature characteristic
                                                handle
    uint16_t        cp_feature_val_hdl;         Cycling power feature characteristic
                                                value handle
    uint8_t         cp_feature_prop;            Cycling power feature characteristic
                                                property
    uint8_t         reserved2;                  Reserved
    uint16_t        sensor_loc_char_hdl;        Sensor location characteristic handle
    uint16_t        sensor_loc _val_hdl;        Sensor location characteristic value
```

```
                                        handle
    uint8_t         sensor_loc _prop;   Sensor location characteristic property
    uint8_t         reserved3;          Reserved
    uint16_t        cp_vector_char_hdl; Cycling power vector characteristic
                                        handle
    uint16_t        cp_vector _val_hdl; Cycling power vector characteristic
                                        value handle
    uint16_t        cp_vector _cfg_hdl; Cycling power vector client
                                        characteristic configuration
                                        descriptor handle
    uint8_t         cp_vector _prop;    Cycling power vector characteristic
                                        property
    uint8_t         reserved4;          Reserved
    uint16_t        cp_cp_char_hdl;     CP control point characteristic handle
    uint16_t        cp_cp _val_hdl;     CP control point characteristic value
                                        handle
    uint16_t        cp_cp _cfg_hdl;     CP control point client characteristic
                                        configuration descriptor handle
    uint8_t         cp_cp _prop;        CP control point characteristic
                                        property
    uint8_t         reserved5;          Reserved
}RBLE_CPS_CONTENT;
```

- Device information service content structures

```
typedef struct RBLE_DIS_CONTENT_t {
    uint16_t        shdl;               Device information service start
                                        handle
    uint16_t        ehdl;               Device information service end
                                        handle
    uint16_t        sys_id_char_hdl;    System ID characteristic handle
    uint16_t        sys_id_val_hdl;     System ID characteristic value
                                        handle
    uint8_t         sys_id_prop;        System ID characteristic property
    uint8_t         reserved;           Reserved
    uint16_t        model_nb_char_hdl;  Model number characteristic handle
    uint16_t        model_nb_val_hdl;   Model number characteristic value
                                        handle
    uint8_t         model_nb_prop;      Model number characteristic
                                        property
    uint8_t         reserved2;          Reserved
    uint16_t        serial_nb_char_hdl; Serial number characteristic handle
    uint16_t        serial_nb_val_hdl;  Serial number characteristic value
                                        handle
    uint8_t         serial_nb_prop;     Serial number characteristic
                                        property
    uint8_t         reserved3;          Reserved
    uint16_t        fw_rev_char_hdl;    Firmware revision characteristic
                                        handle
    uint16_t        fw_rev_val_hdl;     Firmware revision characteristic
                                        value handle
```

```
    uint8_t         fw_rev_prop;                Firmware revision characteristic
                                                property
    uint8_t         reserved4;                  Reserved
    uint16_t        hw_rev_char_hdl;            Hardware revision characteristic
                                                handle
    uint16_t        hw_rev_val_hdl;             Hardware revision characteristic
                                                value handle
    uint8_t         hw_rev_prop;                Hardware revision characteristic
                                                property
    uint8_t         reserved5;                  Reserved
    uint16_t        sw_rev_char_hdl;            Software revision characteristic
                                                handle
    uint16_t        sw_rev_val_hdl;             Software revision characteristic
                                                value handle
    uint8_t         sw_rev_prop;                Software revision characteristic
                                                property
    uint8_t         reserved6;                  Reserved
    uint16_t        manuf_name_char_hdl;        Manufacturer name characteristic
                                                handle
    uint16_t        manuf_name_val_hdl;         Manufacturer name characteristic
                                                value handle
    uint8_t         manuf_name_prop;            Manufacturer name characteristic
                                                property
    uint8_t         reserved7;                  Reserved
    uint16_t        ieee_certif_char_hdl;       IEEE certification characteristic
                                                handle
    uint16_t        ieee_certif_val_hdl;        IEEE certification characteristic
                                                value handle
    uint8_t         ieee_certif_prop;           IEEE certification characteristic
                                                property
    uint8_t         reserved8;                  Reserved
}RBLE_DIS_CONTENT;
```

- Battery service content structures

```
typedef struct RBLE_BATS_CONTENT_t{
    uint16_t        shdl;                       Battery service start handle
    uint16_t        ehdl;                       Battery service end handle
    uint16_t        battery_lvl_char_hdl;       Battery level characteristic handle
    uint16_t        battery_lvl_val_hdl;        Battery level characteristic value
                                                handle
    uint16_t        battery_lvl_cfg_hdl;        Battery level characteristic
                                                configuration descriptor handle
    uint8_t         battery_lvl_prop;           Battery level property
    uint8_t         reserved;                   Reserved
}RBLE_BATS_CONTENT;
```

- CPP Sensor event parameter structures

```
typedef struct RBLE_CPPS_EVENT_t {
    RBLE_CPP_EVENT_TYPE             type;               CPP event type
```

```
uint8_t                         reserved;          Reserved
union Event_Cps_Parameter_u {
    Generic event
    RBLE_STATUS                 status;            Status

    Sensor enable completion event
    struct RBLE_CPP_Sensor_Enable_t{
        RBLE_STATUS             status;            Status
        uint8_t                 reserved;          Reserved
        uint16_t                conhdl;            Connection handle
    }sensor_enable;

    Sensor disable completion event
    struct RBLE_CPP_Sensor_Disable_t{
        uint16_t                conhdl;            Connection handle
        RBLE_CPP_SENSOR_PARAM sensor_info;         Cycling power service information
    }sensor_disable;

    Sensor error indication event
    struct RBLE_CPP_Sensor_Error_Ind_t{
        uint16_t                conhdl;            Connection handle
        RBLE_STATUS             status;            Status
    }error_ind;

    Sensor cycling power measurement value send completion event
    struct RBLE_CPP_Sensor_Send_Measurements_t{
        uint16_t                conhdl;            Connection handle
        RBLE_STATUS             status;            Status
    }send_measurements;

    Sensor cycling power measurement value broadcast completion event
    struct RBLE_CPP_Sensor_Broadcast_Measurements_t{
        uint16_t                conhdl;            Connection handle
        RBLE_STATUS             status;            Status
    }broadcast_measurements;

    Sensor cycling power vector value send completion event
    struct RBLE_CPP_Sensor_Send_Vector_t{
        uint16_t                conhdl;            Connection handle
        RBLE_STATUS             status;            Status
    }send_vector;

    Sensor CP control point send completion event
    struct RBLE_CPP_Sensor_Send_CP_Control_Point_t{
        uint16_t                conhdl;            Connection handle
        RBLE_STATUS             status;            Status
    }send_cp_cp;
```

```
        Sensor battery level send completion event
        struct RBLE_CPP_Sensor_Send_Battery_Level_t{
            uint16_t                conhdl;             Connection handle
            RBLE_STATUS             status;             Status
        }send_battery_level;

        Sensor CP control point change indication event
        struct RBLE_CPP_Sensor_Chg_Cp_Cp_Ind_t{
            uint16_t                conhdl;             Connection handle
            RBLE_CPP_WR_CONTROL_POINT_INFO wr_cp_info;  CP control point setting
                                                        information
        }chg_cp_cp_ind;

        Sensor configuration characteristic value indication event
        struct RBLE_CPP_Sensor_Cfg_indntfbrd_Ind_t{
            uint16_t                conhdl;             Connection handle
            uint8_t                 char_code;          Characteristic value code
            uint8_t                 reserved;           Reserved
            uint16_t                cfg_val;            Configuration characteristic
                                                        value
        }cfg_indntfbrd_ind;

        Sensor command disallowed indication event
        struct RBLE_CPP_Sensor_Command_Disallowed_Ind_t{
            RBLE_STATUS             status;             Status
            uint8_t                 reserved;           Reserved
            uint16_t                opcode;             Opcode
        }cmd_disallowed_ind;
    } param;
} RBLE_CPPS_EVENT;
```

- CPP Collector event parameter structures

```
typedef struct RBLE_CPPC_EVENT_t {
    RBLE_CPP_EVENT_TYPE             type;               CPP event type
    uint8_t                         reserved;           Reserved
    union Event_Cpc_Parameter_u {
        Generic event
        RBLE_STATUS                 status;             Status

        Collector enable completion event
        struct RBLE_CPP_Collector_Enable_t{
            RBLE_STATUS             status;             Status
            uint8_t                 reserved;           Reserved
            uint16_t                conhdl;             Connection handle
            RBLE_CPS_CONTENT        cps;                Cycling power service content
            RBLE_DIS_CONTENT        dis;                Device information service content
```

```
    RBLE_BATS_CONTENT       bas;                Battery service content
}collector_enable;
```

Collector disable completion event

```
struct RBLE_CPP_Collector_Disable_t{
    RBLE_STATUS             status;             Status
    uint8_t                 reserved;           Reserved
    uint16_t                conhdl;             Connection handle
}collector_disable;
```

Collector error indication event

```
struct RBLE_CPP_Collector_Error_Ind_t{
    RBLE_STATUS             status;             Status
    uint8_t                 reserved;           Reserved
    uint16_t                conhdl;             Connection handle
}error_ind;
```

Collector cycling power measurement information notification event

```
struct RBLE_CPP_Collector_Measurements_Ntf_t{
    uint16_t                conhdl;             Connection handle
    RBLE_CPP_MEASUREMENTS_INFO measure_info;    Cycling power measurement
                                                information
}measurements_ntf;
```

Collector cycling power vector information notification event

```
struct RBLE_CPP_Collector_Vector_Ntf_t{
    uint16_t                conhdl;             Connection handle
    RBLE_CPP_VECTOR_INFO    vector_info;        Cycling power vector information
}vector_ntf;
```

Collector CP control point indication event

```
struct RBLE_CPP_Collector_CP_Control_Point_Ind_t{
    uint16_t                conhdl;             Connection handle
    RBLE_CPP_IND_CONTROL_POINT_INFO ind_cp_info;
                                                CP control point information
    uint8_t                 location_num;       Number of stored sensor location
                                                information
    uint8_t                 supported_location[RBLE_CPP_SENSOR_LOCATION_MAX];
                                                Supported sensor location
                                                information
}cp_cp_ind;
```

Collector battery level notification event

```
struct RBLE_CPP_Collector_Battery_Level_Ntf_t{
    uint16_t                conhdl;             Connection handle
    uint8_t                 battery_level;      Battery level
    uint8_t                 reserved;           Reserved
}battery_level_ntf;
```

```
        Collector characteristic value read request response event
        struct RBLE_CPP_Collector_Read_Char_Response_t{
            uint16_t                conhdl;             Connection handle
            uint8_t                 att_code;           Status
            uint8_t                 reserved;           Reserved
            RBLE_ATT_INFO_DATA      data;               Acquired characteristic data
        }rd_char_resp;

        Collector characteristic value write request response event
        struct RBLE_CPP_Collector_Write_Char_Response_t{
            uint16_t                conhdl;             Connection handle
            uint8_t                 att_code;           Status
        }wr_char_resp;

        Collector command disallowed indication event
        struct RBLE_CPP_Collector_Command_Disallowed_Ind_t{
            RBLE_STATUS             status;             Status
            uint8_t                 reserved;           Reserved
            uint16_t                opcode;             Opcode
        }cmd_disallowed_ind;
    } param;
} RBLE_CPPC_EVENT;
```

3.2 Functions

The following table shows the API functions defined for the CPP of rBLE and the following sections describe the API functions in detail.

Table 3-1 API Functions Used by the CPP

RBLE_CPP_Sensor_Enable	Enables the Sensor role.
RBLE_CPP_Sensor_Disable	Disables the Sensor role.
RBLE_CPP_Sensor_Send_Measurements	Sends the cycling power measurement information.
RBLE_CPP_Sensor_Broadcast_Measurements	Broadcasts the cycling power measurement information.
RBLE_CPP_Sensor_Send_Vector	Sends the cycling power vector information.
RBLE_CPP_Sensor_Send_CP_Control_Point	Sends the CP control point information.
RBLE_CPP_Sensor_Send_Battery_Level	Sends the battery level.
RBLE_CPP_Sensor_Send_Write_Response	Sends the write response
RBLE_CPP_Collector_Enable	Enables the Collector role.
RBLE_CPP_Collector_Disable	Disables the Collector role.
RBLE_CPP_Collector_Read_Char	Reads the characteristic value.
RBLE_CPP_Collector_Write_Char	Writes the characteristic value.
RBLE_CPP_Collector_Write_CP_Control_Point	Sets the CP control point.

3.2.1 RBLE_CPP_Sensor_Enable

RBLE_STATUS RBLE_CPP_Sensor_Enable(uint16_t conhdl, uint8_t sec_lvl, uint8_t con_type, RBLE_CPP_SENSOR_PARAM *param, RBLE_CPPS_EVENT_HANDLER call_back)

This function enables the CPP Sensor role.
If the notification/indication/broadcast settings of the transmission data and the sensor location information are configured from the Collector, set the notification/indication/broadcast setting parameter to 0 to configure the connection. If this setting or information has been specified from the Sensor, perform a normal connection in accordance with the notification/indication/broadcast setting parameter and the sensor location information.
The result is reported by using the Sensor role enable completion event RBLE_CPP_EVENT_SENSOR_ENABLE_COMP.

Parameters:

conhdl	Connection handle		
sec_lvl	Security level		
con_type	RBLE_PRF_CON_DISCOVERY	Configuration connection	
	RBLE_PRF_CON_NORMAL	Normal connection	
*param	*cp_meas_ntf_en*	RBLE_PRF_STOP_NTFIND	Stop notification of cycling power measurement information.
		RBLE_PRF_START_NTF	Start notification of cycling power measurement information.
	cp_meas_brd_en	RBLE_PRF_STOP_BRD	Stop broadcast of cycling power measurement information.
		RBLE_PRF_START_BRD	Start broadcast of cycling power measurement information.
	cp_vector_ntf_en	RBLE_PRF_STOP_NTFIND	Stop notification of cycling power vector information.
		RBLE_PRF_START_NTF	Start notification of cycling power vector information.
	cp_cp_ind_en	RBLE_PRF_STOP_NTFIND	Stop indication of CP control point.
		RBLE_PRF_START_IND	Start indication of CP control point.
	battery_level_ntf_en	RBLE_PRF_STOP_NTFIND	Stop notification of battery level information.
		RBLE_PRF_START_NTF	Start notification of battery level information.
	sensor_location	Sensor location information that has been set by the previously connected Collector	
call_back	Specify the callback function that reports the CPP event.		

Return:

RBLE_OK	Success
RBLE_ERR	Error occurred in Sensor role enable processing
RBLE_PARAM_ERR	Invalid parameter
RBLE_STATUS_ERROR	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.

3.2.2 RBLE_CPP_Sensor_Disable

RBLE_STATUS RBLE_CPP_Sensor_Disable(uint16_t conhdl)		
This function disables the CPP Sensor role. The result is reported by using the Sensor role disable completion event RBLE_CPP_EVENT_SENSOR_DISABLE_COMP.		
Parameters:		
conhdl	Connection handle	
Return:		
RBLE_OK	Success	
RBLE_STATUS_ERROR	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.	

3.2.3 RBLE_CPP_Sensor_Send_Measurements

RBLE_STATUS RBLE_CPP_Sensor_Send_Measurements(uint16_t conhdl, RBLE_CPP_MEASUREMENTS_INFO *measurements_info)		
This function sends the measured value data from the sensor. The result is reported by using the Sensor role measured value send completion event RBLE_CPP_EVENT_SENSOR_SEND_MEASUREMENTS_COMP.		
Parameters:		
conhdl	Connection handle	
**measurements_info*	*flags*	Flag that defines whether there is a data field in the characteristic value or not
	instant_power	Instantaneous power
	pedal_balance	Pedal power balance
	accumulated_torque	Accumulated torque
	wheel_revolutions	Cumulative wheel revolutions
	wheel_event	Last wheel event time
	crank_revolutions	Cumulative crank revolutions
	crank_event	Last crank event time
	max_force_magnitude	Maximum force magnitude
	min_force_magnitude	Minimum force magnitude
	max_torque_magnitude	Maximum torque magnitude
	min_torque_magnitude	Minimum torque magnitude
	max_angle	Maximum angle
	min_angle	Minimum angle
	top_dead_spot	Top dead spot angle
	bottom_dead_spot	Bottom dead spot angle
	accumulated_energy	Accumulated energy
Return:		
RBLE_OK	Success	
RBLE_STATUS_ERROR	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.	

3.2.4 RBLE_CPP_Sensor_Broadcast_Measurements

RBLE_STATUS RBLE_CPP_Sensor_Broadcast_Measurements(uint16_t conhdl,
RBLE_CPP_MEASUREMENTS_INFO *measurements_info)

This function broadcasts the measured value data from the sensor.
The result is reported by using the Sensor role measured value broadcast completion event RBLE_CPP_EVENT_SENSOR_BROADCAST_MEASUREMENTS_COMP.

Parameters:

conhdl	Connection handle	
*measurements_info	*flags*	Flag that defines whether there is a data field in the characteristic value or not
	instant_power	Instantaneous power
	pedal_balance	Pedal power balance
	accumulated_torque	Accumulated torque
	wheel_revolutions	Cumulative wheel revolutions
	wheel_event	Last wheel event time
	crank_revolutions	Cumulative crank revolutions
	crank_event	Last crank event time
	max_force_magnitude	Maximum force magnitude
	min_force_magnitude	Minimum force magnitude
	max_torque_magnitude	Maximum torque magnitude
	min_torque_magnitude	Minimum torque magnitude
	max_angle	Maximum angle
	min_angle	Minimum angle
	top_dead_spot	Top dead spot angle
	bottom_dead_spot	Bottom dead spot angle
	accumulated_energy	Accumulated energy

Return:

RBLE_OK	Success
RBLE_STATUS_ERROR	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.

3.2.5 RBLE_CPP_Sensor_Send_Vector

RBLE_STATUS RBLE_CPP_Sensor_Send_Vector(uint16_t conhdl, RBLE_CPP_VECTOR_INFO *vector_info)		
This function sends the cycling power vector data from the sensor. Set the information of either force [N] or torque [1/32 Nm] to the magnitude[ ], and set the flags (force: bit[2]=1 / torque: bit[3]=1) according to the information that have set up. The result is reported by using the Sensor role cycling power vector send completion event RBLE_CPP_EVENT_SENSOR_SEND_VECTOR_COMP.		
Parameters:		
conhdl	Connection handle	
**vector_info*	*flags*	Flag that defines whether there is a data field in the characteristic value or not
	crank_revolutions	Cumulative crank revolutions
	crank_event	Last crank event time
	first_crank_angle	First crank measurement angle
	array_num	Number of magnitude information
	magnitude[RBLE_CPP_MAGNITUDE_MAX]	Instantaneous Force Magnitude Array or Instantaneous Torque Magnitude Array
Return:		
RBLE_OK	Success	
RBLE_STATUS_ERROR	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.	

3.2.6 RBLE_CPP_Sensor_Send_CP_Control_Point

RBLE_STATUS RBLE_CPP_Sensor_Send_CP_Control_Point (uint16_t conhdl,
RBLE_CPP_IND_CONTROL_POINT_INFO *ind_cp_info)

This function sends the CP control point information from the sensor.
If the operation is written to the CP control point from Collector, set RBLE_CPP_OP_RESPONSE_CODE to OpeCode.
Set the operation code that is sent from the Collector to the request_op_code, set the status for the operation to the response_value.
Depending on the operation that is requested by the Collector, set the additional parameters required.
In addition, if the requested operation is RBLE_CPP_OP_REQ_SUPPORTED_SL_CODE, the supported sensor location information is set as the initial value is sent.
The result is reported by using the Sensor role CP control point send completion event RBLE_CPP_EVENT_SENSOR_SEND_CP_CP_COMP.

Parameters:

conhdl	Connection handle		
**ind_cp_info*	*OpCode*	RBLE_CPP_OP_RESPONSE_CODE	Response code
	request_op_code	RBLE_CPP_OP_SET_CUMULATIVE_CODE	Set cumulative value
		RBLE_CPP_OP_UPDATE_SL_CODE	Update sensor location
		RBLE_CPP_OP_REQ_SUPPORTED_SL_CODE	Request supported sensor locations
		RBLE_CPP_OP_SET_CRANK_LEN_CODE	Set crank length
		RBLE_CPP_OP_REQ_CRANK_LEN_CODE	Request crank length (set the crank_length)
		RBLE_CPP_OP_SET_CHAIN_LEN_CODE	Set chain length
		RBLE_CPP_OP_REQ_CHAIN_LEN_CODE	Request chain length (set the chain_length)
		RBLE_CPP_OP_SET_CHAIN_WEI_CODE	Set chain weight
		RBLE_CPP_OP_REQ_CHAIN_WEI_CODE	Request chain weight (set the chain_weight)
		RBLE_CPP_OP_SET_SPAN_LEN_CODE	Set span length
		RBLE_CPP_OP_REQ_SPAN_LEN_CODE	Request span length (set the span_length)
		RBLE_CPP_OP_START_OFFSET_COMPENSATION_CODE	Start offset compensation (set the offset_compensation)
		RBLE_CPP_OP_MASK_CP_MEAS_CONTENT_CODE	Mask cycling power measurement characteristic content
		RBLE_CPP_OP_REQ_SAMPL_RATE_CODE	Request sampling rate (set the sampling_rate)
		RBLE_CPP_OP_REQ_FACTORY_CALIB_DATE_CODE	Request factory calibration date (set the stamp)

RBLE_STATUS RBLE_CPP_Sensor_Send_CP_Control_Point (uint16_t conhdl, RBLE_CPP_IND_CONTROL_POINT_INFO *ind_cp_info)				
		reqponse_value	RBLE_CPP_RES_SUCCESS_CODE	*Success*
			RBLE_CPP_RES_NOT_SUPPORTED_CODE	*Op Code not supported*
			RBLE_CPP_RES_INVALID_PARAM_CODE	*Invalid parameter*
			RBLE_CPP_RES_OP_FAILED_CODE	*Operation failed*
		crank_length	Crank length value	
		chain_length	Chain length value	
		chain_weight	Chain weight value	
		span_length	Span length value	
		offset_compensation	Offset compensation	
		sampling_rate	Sampling rate	
		stamp	*year*	Year
			month	Month
			day	Day
			hour	Hour
			min	Minute
			sec	Second
Return:				
	RBLE_OK	Success		
	RBLE_STATUS_ERROR	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.		

3.2.7 RBLE_CPP_Sensor_Send_Battery_Level

RBLE_STATUS RBLE_CPP_Sensor_Send_Battery_Level(uint16_t conhdl, uint8_t battery_level)	
This function sends the battery level of the sensor. The result is reported by using the Sensor role battery level send completion event RBLE_CPP_EVENT_SENSOR_SEND_BATTERY_LEVEL_COMP.	
Parameters:	
conhdl	Connection handle
battery_level	Battery level
Return:	
RBLE_OK	Success
RBLE_STATUS_ERROR	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.

3.2.8 RBLE_CPP_Sensor_Send_Write_Response

RBLE_STATUS RBLE_CPP_Sensor_Send_Write_Response(uint16_t conhdl, uint8_t res_code)			
This function sends response to write request of Cycling Power Vector Client characteristic configuration descriptor. When write request to Cycling Power Vector client configuration descriptor (RBLE_CPP_EVENT_SENSOR_CFG_INDNTFBRD_IND event) is sent by Collector, send response to the Collector by using this function. If the current connection parameters do not allow the sending of notification (e.g., the Server requires faster connection interval), Sensor should requests new connection parameter to Collector. RBLE_ATT_ERR_APP_ERROR should be set to res_code to respond to Collector if the Collector did not change the connection parameters. Otherwise RBLE_ATT_ERR_NO_ERROR should be set to res_code to respond to Collector.			
Parameters:			
	conhdl	Connection handle	
	res_code	RBLE_ATT_ERR_NO_ERROR	Success
		RBLE_ATT_ERR_APP_ERROR	Application error
		See *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for ATT error code*.	
Return:			
	RBLE_OK	Success	
	RBLE_STATUS_ERROR	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.	

3.2.9 RBLE_CPP_Collector_Enable

RBLE_STATUS RBLE_CPP_Collector_Enable(uint16_t conhdl, uint8_t con_type,
RBLE_CPS_CONTENT *cps, RBLE_DIS_CONTENT *dis, RBLE_BATS_CONTENT *bas,
RBLE_CPPC_EVENT_HANDLER call_back)

This function enables the CPP Collector role and starts access to the service exposed by the CPP Sensor. The result is reported by using the Collector role enable completion event RBLE_CPP_EVENT_COLLECTOR_ENABLE_COMP.
When starting access to the service exposed by a Sensor to be connected for the first time, set 0 to the parameters of the service to configure the connection and to discover the service for the Sensor. If the handle information about the discovered service is saved and is used when the Sensor is connected normally for a second or subsequent time, detecting the service is skipped, which enables a high-speed access to the service.
While the Collector role is enabled, the service exposed by only one Sensor is accessible. To connect to more than one Sensor at the same time and access the services exposed by each Sensor, repeat enable/disable of the Collector role in order to switch access to them. At that time, perform normal connection by using the connection handle (which was obtained when connecting to each Sensor) and the handle information (which was saved when starting access to the service for the first time) as parameters.

Parameters:

conhdl	Connection handle	
con_type	RBLE_PRF_CON_DISCOVERY	Configuration connection performed when connecting for the first time
	RBLE_PRF_CON_NORMAL	Normal connection performed when connecting for the second and subsequent times
**cps*	*shdl*	Cycling power service start handle
	ehdl	Cycling power service end handle
	cp_meas_char_hdl	Cycling power measurement characteristic handle
	cp_meas_val_hdl	Cycling power measurement characteristic value handle
	cp_meas_cfg_hdl	Cycling power measurement client characteristic configuration descriptor handle
	cp_meas_cfg_brd_hdl	Cycling power measurement server characteristic configuration descriptor handle
	cp_meas_prop	Cycling power measurement characteristic property
	cp_feature_char_hdl	Cycling power feature characteristic handle
	cp_feature_val_hdl	Cycling power feature characteristic value handle
	cp_feature_prop	Cycling power feature characteristic property
	sensor_loc_char_hdl	Sensor location characteristic handle
	sensor_loc_val_hdl	Sensor location characteristic value handle
	sensor_loc_prop	Sensor location characteristic property
	cp_vector_char_hdl	Cycling power vector characteristic handle
	cp_vector_val_hdl	Cycling power vector characteristic value handle
	cp_vector_cfg_hdl	Cycling power vector client characteristic configuration descriptor handle
	cp_vector_prop	Cycling power vector characteristic property
	cp_cp_char_hdl	CP control point characteristic handle
	cp_cp_val_hdl	CP control point characteristic value handle
	cp_cp_cfg_hdl	CP control point client characteristic configuration descriptor handle
	cp_cp_prop	CP control point characteristic property

	*dis	*shdl*	Device information service start handle
		ehdl	Device information service end handle
		sys_id_char_hdl	System ID characteristic handle
		sys_id_val_hdl	System ID characteristic value handle
		sys_id_prop	System ID characteristic property
		model_nb_char_hdl	Model number characteristic handle
		model_nb_val_hdl	Model number characteristic value handle
		model_nb_prop	Model number characteristic property
		serial_nb_char_hdl	Serial number characteristic handle
		serial_nb_val_hdl	Serial number characteristic value handle
		serial_nb_prop	Serial number characteristic property
		fw_rev_char_hdl	Firmware revision characteristic handle
		fw_rev_val_hdl	Firmware revision characteristic value handle
		fw_rev_prop	Firmware revision characteristic property
		hw_rev_char_hdl	Hardware revision characteristic handle
		hw_rev_val_hdl	Hardware revision characteristic value handle
		hw_rev_prop	Hardware revision characteristic property
		sw_rev_char_hdl	Software revision characteristic handle
		sw_rev_val_hdl	Software revision characteristic value handle
		sw_rev_prop	Software revision characteristic property
		manuf_name_char_hdl	Manufacturer name characteristic handle
		manuf_name_val_hdl	Manufacturer name characteristic value handle
		manuf_name_prop	Manufacturer name characteristic property
		ieee_certif_char_hdl	IEEE certification characteristic handle
		ieee_certif_val_hdl	IEEE certification characteristic value handle
		ieee_certif_prop	IEEE certification characteristic property
	*bas	*shdl*	Battery service start handle
		ehdl	Battery service end handle
		battery_lvl_char_hdl	Battery Level characteristic handle
		battery_lvl_val_hdl	Battery Level characteristic value handle
		battery_lvl_cfg_hdl	Battery Level characteristic configuration descriptor handle
		battery_lvl_prop	Battery Level property
	call_back	Callback	

Return:

RBLE_OK	Success
RBLE_ERR	Error occurred in initialization processing
RBLE_PARAM_ERR	Invalid parameter
RBLE_STATUS_ERROR	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.

3.2.10 RBLE_CPP_Collector_Disable

RBLE_STATUS RBLE_CPP_Collector_Disable(uint16_t conhdl)		
This function disables the CPP Collector role and terminates the access to the service exposed by CPP Sensor. The result is reported by using the Collector role disable completion event RBLE_CPP_EVENT_COLLECTOR_DISABLE_COMP.		
Parameters:		
conhdl	Connection handle	
Return:		
RBLE_OK	Success	
RBLE_STATUS_ERROR	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.	

3.2.11 RBLE_CPP_Collector_Read_Char

RBLE_STATUS RBLE_CPP_Collector_Read_Char (uint16_t conhdl, uint8_t char_code)		
This function reads the characteristic value of the cycling power service, the device information service and the battery service. The result is reported by using the characteristic value read request response event RBLE_CPP_EVENT_COLLECTOR_READ_CHAR_RESPONSE.		
Parameters:		
conhdl	Connection handle	
char_code	RBLE_CPPC_RD_CPS_CM_CFG	Cycling power measurement notification
	RBLE_CPPC_RD_CPS_CM_BRD_CFG	Cycling power measurement broadcast
	RBLE_CPPC_RD_CPS_CV_CFG	Cycling power vector notification
	RBLE_CPPC_RD_CPS_CPCP_CFG	CP control point indication
	RBLE_CPPC_RD_CPS_CP_FEATURE	Cycling power feature
	RBLE_CPPC_RD_CPS_SL	Cy**c**ling power sensor location
	RBLE_CPPC_RD_DIS_MANUF	Sensor manufacturer name
	RBLE_CPPC_RD_DIS_MODEL	Sensor model number
	RBLE_CPPC_RD_DIS_SERNB	Sensor serial number
	RBLE_CPPC_RD_DIS_HWREV	Sensor hardware revision
	RBLE_CPPC_RD_DIS_FWREV	Sensor firmware revision
	RBLE_CPPC_RD_DIS_SWREV	Sensor software revision
	RBLE_CPPC_RD_DIS_SYSID	Sensor system ID
	RBLE_CPPC_RD_DIS_IEEE	Sensor IEEE certification information
	RBLE_CPPC_RD_BAS_BL	Battery level
	RBLE_CPPC_RD_BAS_BL_CFG	Battery level notification
Return:		
RBLE_OK	Success	
RBLE_STATUS_ERROR	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.	

3.2.12 RBLE_CPP_Collector_Write_Char

RBLE_STATUS RBLE_CPP_Collector_Write_Char(uint16_t conhdl, uint8_t char_code, uint16_t cfg_val)			
This function writes each client/server characteristic configuration descriptor of the cycling power service or the battery service. The result is reported by using the characteristic value write request response event RBLE_CPP_EVENT_COLLECTOR_WRITE_CHAR_RESPONSE.			
Parameters:			
	conhdl	Connection handle	
	char_code	RBLE_CPPC_CYCPWR_MEAS_CODE	Cycling power measurement client characteristic configuration descriptor
		RBLE_CPPC_CYCPWR_MEAS_BRD_CODE	Cycling power measurement server characteristic configuration descriptor
		RBLE_CPPC_CYCPWR_VCTR_CODE	Cycling power vector client characteristic configuration descriptor
		RBLE_CPPC_CYCPWR_CONTROL_POINT_CODE	CP control point client characteristic configuration descriptor
		RBLE_CPPC_BATTERY_LEVEL_CODE	Battery level client characteristic configuration descriptor
	cfg_val	RBLE_PRF_STOP_NTFIND	Stop notification/ indication.
		RBLE_PRF_START_NTF	Start notification.
		RBLE_PRF_START_IND	Start indication.
		RBLE_PRF_STOP_BRD	Stop Broadcast.
		RBLE_PRF_START_BRD	Start Broadcast.
Return:			
	RBLE_OK	Success	
	RBLE_STATUS_ERROR	Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.	

3.2.13 RBLE_CPP_Collector_Write_CP_Control_Point

RBLE_STATUS RBLE_CPP_Collector_Write_CP_Control_Point(uint16_t conhdl, RBLE_CPP_WR_CONTROL_POINT_INFO * wr_cp_info)				
This function sets the CP control point characteristic information of the cycling power service. If OpCode requires parameters, set the additional parameters to be set according to the operation. The result is reported by using the characteristic value write request response event RBLE_CPP_EVENT_COLLECTOR_WRITE_CHAR_RESPONSE.				
Parameters:				
	conhdl	Connection handle		
	**wr_cp_info*	CP control point setting value		
		OpCode	RBLE_CPP_OP_SET_CUMULATIVE_CODE	Set cumulative value (set the cumulative_value)
			RBLE_CPP_OP_UPDATE_SL_CODE	Update sensor location (set the sensor_location)
			RBLE_CPP_OP_REQ_SUPPORTED_SL_CODE	Request supported sensor locations
			RBLE_CPP_OP_SET_CRANK_LEN_CODE	Set crank length (set the crank_length)
			RBLE_CPP_OP_REQ_CRANK_LEN_CODE	Request crank length
			RBLE_CPP_OP_SET_CHAIN_LEN_CODE	Set chain length (set the chain_length)
			RBLE_CPP_OP_REQ_CHAIN_LEN_CODE	Request chain length
			RBLE_CPP_OP_SET_CHAIN_WEI_CODE	Set chain weight (set the chain_weight)
			RBLE_CPP_OP_REQ_CHAIN_WEI_CODE	Request chain weight
			RBLE_CPP_OP_SET_SPAN_LEN_CODE	Set span length (set the span_length)
			RBLE_CPP_OP_REQ_SPAN_LEN_CODE	Request span length
			RBLE_CPP_OP_START_OFFSET_COMPENSATION_CODE	Start offset compensation
			RBLE_CPP_OP_MASK_CP_MEAS_CONTENT_CODE	Mask cycling power measurement characteristic content (set the mask_meas_content)
			RBLE_CPP_OP_REQ_SAMPL_RATE_CODE	Request sampling rate
			RBLE_CPP_OP_REQ_FACTORY_CALIB_DATE_CODE	Request factory calibration date
		cumulative_value	Cumulative value	
		sensor_location	RBLE_CPPC_SENSOR_OTHER	Other
			RBLE_CPPC_SENSOR_TOP_OF_SHOE	Top of shoe
			RBLE_CPPC_SENSOR_IN_SHOE	In shoe
			RBLE_CPPC_SENSOR_HIP	Hip

RBLE_STATUS RBLE_CPP_Collector_Write_CP_Control_Point(uint16_t conhdl, RBLE_CPP_WR_CONTROL_POINT_INFO * wr_cp_info)				
			RBLE_CPPC_SENSOR_FRONT_WHEEL	Front wheel
			RBLE_CPPC_SENSOR_LEFT_CRANK	Left crank
			RBLE_CPPC_SENSOR_RIGHT_CRANK	Right crank
			RBLE_CPPC_SENSOR_LEFT_PEDAL	Left pedal
			RBLE_CPPC_SENSOR_RIGHT_PEDAL	Right pedal
			RBLE_CPPC_SENSOR_FRONT_HUB	Front hub
			RBLE_CPPC_SENSOR_REAR_DROPOUT	Rear dropout
			RBLE_CPPC_SENSOR_CHAINSTAY	Chainstay
			RBLE_CPPC_SENSOR_REAR_WHEEL	Rear wheel
			RBLE_CPPC_SENSOR_REAR_HUB	Rear hub
			RBLE_CPPC_SENSOR_CHEST	Chest
		crank_length	Crank length	
		chain_length	Chain length	
		chain_weight	Chain weight	
		Span_length	Span length	
		Mask_meas_content	Mask setting of cycling power measurement characteristic content	
Return:				
	RBLE_OK		Success	
	RBLE_STATUS_ERROR		Not executable because the rBLE mode is other than RBLE_MODE_ACTIVE.	

3.3 Events

The following table shows the events defined for the CPP of rBLE and the following sections describe the events in detail.

Table 3-2 Events Defined for the CPP

RBLE_CPP_EVENT_SENSOR_ENABLE_COMP	Sensor role enable completion event
RBLE_CPP_EVENT_SENSOR_DISABLE_COMP	Sensor role disable completion event
RBLE_CPP_EVENT_SENSOR_ERROR_IND	Sensor role error indication event
RBLE_CPP_EVENT_SENSOR_SEND_MEASUREMENTS_COMP	Cycling power measurement information send completion event
RBLE_CPP_EVENT_SENSOR_BROADCAST_MEASUREMENTS_COMP	Cycling power measurement information broadcast completion event
RBLE_CPP_EVENT_SENSOR_SEND_VECTOR_COMP	Cycling power vector information send completion event
RBLE_CPP_EVENT_SENSOR_SEND_CP_CP_COMP	CP control point information send completion event
RBLE_CPP_EVENT_SENSOR_SEND_BATTERY_LEVEL_COMP	Battery level information send completion event
RBLE_CPP_EVENT_SENSOR_CHG_CP_CP_IND	CP control point change indication event
RBLE_CPP_EVENT_SENSOR_CFG_INDNTFBRD_IND	Characteristic value indication event
RBLE_CPP_EVENT_SENSOR_COMMAND_DISALLOWED_IND	Sensor role command disallowed indication event
RBLE_CPP_EVENT_COLLECTOR_ENABLE_COMP	Collector role enable completion event
RBLE_CPP_EVENT_COLLECTOR_DISABLE_COMP	Collector role disable completion event
RBLE_CPP_EVENT_COLLECTOR_ERROR_IND	Collector role error indication event
RBLE_CPP_EVENT_COLLECTOR_MEASUREMENTS_NTF	Cycling power measurement notification event
RBLE_CPP_EVENT_COLLECTOR_VECTOR_NTF	Cycling power vector notification event
RBLE_CPP_EVENT_COLLECTOR_CP_CP_IND	CP control point indication event
RBLE_CPP_EVENT_COLLECTOR_BATTERY_LEVEL_NTF	Battery level notification event
RBLE_CPP_EVENT_COLLECTOR_READ_CHAR_RESPONSE	Characteristic value read request response event
RBLE_CPP_EVENT_COLLECTOR_WRITE_CHAR_RESPONSE	Characteristic value write request response event
RBLE_CPP_EVENT_COLLECTOR_COMMAND_DISALLOWED_IND	Collector role command disallowed indication event

3.3.1 RBLE_CPP_EVENT_SENSOR_ENABLE_COMP

RBLE_CPP_EVENT_SENSOR_ENABLE_COMP	
This event reports the result of enabling the Sensor role (RBLE_CPP_Sensor_Enable).	
Parameters:	
status	Result of enabling the Sensor role (See *2.2* and *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for rBLE status*.)
conhdl	Connection handle

3.3.2 RBLE_CPP_EVENT_SENSOR_DISABLE_COMP

RBLE_CPP_EVENT_SENSOR_DISABLE_COMP			
This event reports the result of disabling the Sensor role (RBLE_CPP_Sensor_Disable).			
Parameters:			
conhdl	Connection handle		
sensor_info	*cp_meas_ntf_en*	RBLE_PRF_STOP_NTFIND	Stop notification of cycling power measurement information.
		RBLE_PRF_START_NTF	Start notification of cycling power measurement information.
	cp_meas_brd_en	RBLE_PRF_STOP_BRD	Stop broadcast of cycling power measurement information.
		RBLE_PRF_START_BRD	Start broadcast of cycling power measurement information.
	cp_vector_ntf_en	RBLE_PRF_STOP_NTFIND	Stop notification of cycling power vector information.
		RBLE_PRF_START_NTF	Start notification of cycling power vector information.
	cp_cp_ind_en	RBLE_PRF_STOP_NTFIND	Stop indication of CP control point.
		RBLE_PRF_START_IND	Start indication of CP control point.
	battery_level_ntf_en	RBLE_PRF_STOP_NTFIND	Stop notification of battery level information.
		RBLE_PRF_START_NTF	Start notification of battery level information.
	sensor_location	Sensor location information that has been set by the Collector	

3.3.3 RBLE_CPP_EVENT_SENSOR_ERROR_IND

RBLE_CPP_EVENT_SENSOR_ERROR_IND		
This event indicates an error code unique to the Sensor role.		
Parameters:		
	conhdl	Connection handle
	status	Error code (See *2.2* and *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for rBLE status*.)

3.3.4 RBLE_CPP_EVENT_SENSOR_SEND_MEASUREMENTS_COMP

RBLE_CPP_EVENT_SENSOR_SEND_MEASUREMENTS_COMP		
This event reports completion of sending the measured value (RBLE_CPP_Sensor_Send_Measurements).		
Parameters:		
	conhdl	Connection handle
	status	Measured value send completion result (See *2.2* and *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for rBLE status*.)

3.3.5 RBLE_CPP_EVENT_SENSOR_BROADCAST_MEASUREMENTS_COMP

RBLE_CPP_EVENT_SENSOR_BROADCAST_MEASUREMENTS_COMP		
This event reports completion of broadcasting the measured value (RBLE_CPP_Sensor_Broadcast_Measurements).		
Parameters:		
	conhdl	Connection handle
	status	Measured value broadcast completion result (See *2.2* and *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for rBLE status*.)

3.3.6 RBLE_CPP_EVENT_SENSOR_SEND_VECTOR_COMP

RBLE_CPP_EVENT_SENSOR_SEND_VECTOR_COMP		
This event reports completion of sending the cycling power vector value (RBLE_CPP_Sensor_Send_Vector).		
Parameters:		
	conhdl	Connection handle
	status	Cycling power vector value send completion result (See *2.2* and *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for rBLE status*.)

3.3.7 RBLE_CPP_EVENT_SENSOR_SEND_CP_CP_COMP

RBLE_CPP_EVENT_SENSOR_SEND_CP_CP_COMP		
This event reports completion of sending the CP control point information (RBLE_CPP_Sensor_Send_CP_Control_Point).		
Parameters:		
	conhdl	Connection handle
	status	CP control point information send completion result (See *2.2* and *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for rBLE status*.)

3.3.8 RBLE_CPP_EVENT_SENSOR_SEND_BATTERY_LEVEL_COMP

RBLE_CPP_EVENT_SENSOR_SEND_BATTERY_LEVEL_COMP		
This event reports completion of sending the battery level (RBLE_CPP_Sensor_Send_Battery_Level).		
Parameters:		
	conhdl	Connection handle
	status	Battery level send completion result (See *2.2* and *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for rBLE status*.)

3.3.9 RBLE_CPP_EVENT_SENSOR_CHG_CP_CP_IND

RBLE_CPP_EVENT_SENSOR_CHG_CP_CP_IND			
This event indicates that the value of the CP control point characteristic of the cycling power service has been changed by the Collector. If additional parameter is present, the parameter corresponding to the OpCode is set.			
Parameters:			
conhdl	Connection handle		
wr_cp_info	CP control point setting value		
	OpCode	RBLE_CPP_OP_SET_CUMULATIVE_CODE	Set cumulative value (reference to the cumulative_value)
		RBLE_CPP_OP_UPDATE_SL_CODE	Update sensor location (reference to the sensor_location)
		RBLE_CPP_OP_REQ_SUPPORTED_SL_CODE	Request supported sensor locations
		RBLE_CPP_OP_SET_CRANK_LEN_CODE	Set crank length (reference to the crank_length)
		RBLE_CPP_OP_REQ_CRANK_LEN_CODE	Request crank length
		RBLE_CPP_OP_SET_CHAIN_LEN_CODE	Set chain length (reference to the chain_length)
		RBLE_CPP_OP_REQ_CHAIN_LEN_CODE	Request chain length
		RBLE_CPP_OP_SET_CHAIN_WEI_CODE	Set chain weight (reference to the chain_weight)
		RBLE_CPP_OP_REQ_CHAIN_WEI_CODE	Request chain weight
		RBLE_CPP_OP_SET_SPAN_LEN_CODE	Set span length (reference to the span_length)
		RBLE_CPP_OP_REQ_SPAN_LEN_CODE	Request span length
		RBLE_CPP_OP_START_OFFSET_COMPENSATION_CODE	Start offset compensation
		RBLE_CPP_OP_MASK_CP_MEAS_CONTENT_CODE	Mask cycling power measurement characteristic content (reference to the mask_meas_content)
		RBLE_CPP_OP_REQ_SAMPL_RATE_CODE	Request sampling rate
		RBLE_CPP_OP_REQ_FACTORY_CALIB_DATE_CODE	Request factory calibration date
	cumulative_value	Cumulative value	
	sensor_location	RBLE_CPPC_SENSOR_OTHER	Other
		RBLE_CPPC_SENSOR_TOP_OF_SHOE	Top of shoe

RBLE_CPP_EVENT_SENSOR_CHG_CP_CP_IND				
			RBLE_CPPC_SENSOR_IN_SHOE	In shoe
			RBLE_CPPC_SENSOR_HIP	Hip
			RBLE_CPPC_SENSOR_FRONT_WHEEL	Front wheel
			RBLE_CPPC_SENSOR_LEFT_CRANK	Left crank
			RBLE_CPPC_SENSOR_RIGHT_CRANK	Right crank
			RBLE_CPPC_SENSOR_LEFT_PEDAL	Left pedal
			RBLE_CPPC_SENSOR_RIGHT_PEDAL	Right pedal
			RBLE_CPPC_SENSOR_FRONT_HUB	Front hub
			RBLE_CPPC_SENSOR_REAR_DROPOUT	Rear dropout
			RBLE_CPPC_SENSOR_CHAINSTAY	Chainstay
			RBLE_CPPC_SENSOR_REAR_WHEEL	Rear wheel
			RBLE_CPPC_SENSOR_REAR_HUB	Rear hub
			RBLE_CPPC_SENSOR_CHEST	Chest
		crank_length	Crank length	
		chain_length	Chain length	
		chain_weight	Chain weight	
		Span_length	Span length	
		Mask_meas_content	Mask setting of cycling power measurement characteristic content	

3.3.10 RBLE_CPP_EVENT_SENSOR_CFG_INDNTFBRD_IND

RBLE_CPP_EVENT_SENSOR_CFG_INDNTFBRD_IND		
This event indicates that the value of the client/server characteristic configuration descriptor of the cycling power service or the battery service has been set by the Collector. When write request to client characteristic configuration descriptor of Cycling Power Vector is sent by the Collector(char_code = RBLE_CPPC_CYCPWR_VCTR_CODE).sends response to the Collector by using the RBLE_CPP_Sensor_Send_Write_Response function. When write request to client characteristic configuration descriptor of other characteristic is sent by the Collector, response is automatically sent to the Collector.		
Parameters:		
conhdl	Connection handle	
char_code	RBLE_CPPC_CYCPWR_MEAS_CODE	Cycling power measurement client characteristic configuration descriptor
	RBLE_CPPC_CYCPWR_MEAS_BRD_CODE	Cycling power measurement server characteristic configuration descriptor
	RBLE_CPPC_CYCPWR_VCTR_CODE	Cycling power vector client characteristic configuration descriptor
	RBLE_CPPC_CYCPWR_CONTROL_POINT_CODE	CP control point client characteristic configuration descriptor
	RBLE_CPPC_BATTERY_LEVEL_CODE	Battery level client characteristic configuration descriptor
cfg_val	RBLE_PRF_STOP_NTFIND	Stop notification/ indication.
	RBLE_PRF_START_NTF	Start notification/broadcast.
	RBLE_PRF_START_IND	Start indication.
	RBLE_PRF_STOP_BRD	Stop broadcast.
	RBLE_PRF_START_BRD	Start broadcast.

3.3.11 RBLE_CPP_EVENT_SENSOR_COMMAND_DISALLOWED_IND

RBLE_CPP_EVENT_SENSOR_COMMAND_DISALLOWED_IND		
This event indicates the error that occurs when a command executed by the Sensor role cannot be accepted.		
Parameters:		
status	Result of command execution (See *2.2* and *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for rBLE status*.)	
opcode	RBLE_CMD_CPP_SENSOR_ENABLE	Sensor role enable command
	RBLE_CMD_CPP_SENSOR_DISABLE	Sensor role disable command
	RBLE_CMD_CPP_SENSOR_SEND_MEASUREMENTS	Cycling power measurement send command
	RBLE_CMD_CPP_SENSOR_BROADCAST_MEASUREMENTS	Cycling power measurement broadcast command
	RBLE_CMD_CPP_SENSOR_SEND_VECTOR	Cycling power vector send command
	RBLE_CMD_CPP_SENSOR_SEND_CONTROL_POINT	CP control point send command
	RBLE_CMD_CPP_SENSOR_SEND_BATTERY_LEVEL	Battery level send command

3.3.12 RBLE_CPP_EVENT_COLLECTOR_ENABLE_COMP

RBLE_CPP_EVENT_COLLECTOR_ENABLE_COMP		
This event reports the result of enabling the Collector role (RBLE_CPP_Collector_Enable). Save the obtained handle information about the discovered service, to enable a high-speed access to the service without service detection when restarting access to the service.		
Parameters:		
status	Result of command execution (See *2.2* and *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for rBLE status*.)	
conhdl	Connection handle	
cps	*shdl*	Cycling power service start handle
	ehdl	Cycling power service end handle
	cp_meas_char_hdl	Cycling power measurement characteristic handle
	cp_meas_val_hdl	Cycling power measurement characteristic value handle
	cp_meas_cfg_hdl	Cycling power measurement client characteristic configuration descriptor handle
	cp_meas_cfg_brd_hdl	Cycling power measurement server characteristic configuration descriptor handle
	cp_meas_prop	Cycling power measurement characteristic property
	cp_feature_char_hdl	Cycling power feature characteristic handle
	cp_feature_val_hdl	Cycling power feature characteristic value handle
	cp_feature_prop	Cycling power feature characteristic property
	sensor_loc_char_hdl	Sensor location characteristic handle
	sensor_loc_val_hdl	Sensor location characteristic value handle
	sensor_loc_prop	Sensor location characteristic property
	cp_vector_char_hdl	Cycling power vector characteristic handle
	cp_vector_val_hdl	Cycling power vector characteristic value handle
	cp_vector_cfg_hdl	Cycling power vector client characteristic configuration descriptor handle
	cp_vector_prop	Cycling power vector characteristic property
	cp_cp_char_hdl	CP control point characteristic handle
	cp_cp_val_hdl	CP control point characteristic value handle
	cp_cp_cfg_hdl	CP control point client characteristic configuration descriptor handle
	cp_cp_prop	CP control point characteristic property
dis	*shdl*	Device information service start handle
	ehdl	Device information service end handle
	sys_id_char_hdl	System ID characteristic handle
	sys_id_val_hdl	System ID characteristic value handle
	sys_id_prop	System ID characteristic property
	model_nb_char_hdl	Model number characteristic handle
	model_nb_val_hdl	Model number characteristic value handle
	model_nb_prop	Model number characteristic property
	serial_nb_char_hdl	Serial number characteristic handle
	serial_nb_val_hdl	Serial number characteristic value handle

		serial_nb_prop	Serial number characteristic property
		fw_rev_char_hdl	Firmware revision characteristic handle
		fw_rev_val_hdl	Firmware revision characteristic value handle
		fw_rev_prop	Firmware revision characteristic property
		hw_rev_char_hdl	Hardware revision characteristic handle
		hw_rev_val_hdl	Hardware revision characteristic value handle
		hw_rev_prop	Hardware revision characteristic property
		sw_rev_char_hdl	Software revision characteristic handle
		sw_rev_val_hdl	Software revision characteristic value handle
		sw_rev_prop	Software revision characteristic property
		manuf_name_char_hdl	Manufacturer name characteristic handle
		manuf_name_val_hdl	Manufacturer name characteristic value handle
		manuf_name_prop	Manufacturer name characteristic property
		ieee_certif_char_hdl	IEEE certification characteristic handle
		ieee_certif_val_hdl	IEEE certification characteristic value handle
		ieee_certif_prop	IEEE certification characteristic property
	bas	*shdl*	Battery service start handle
		ehdl	Battery service end handle
		battery_lvl_char_hdl	Battery Level characteristic handle
		battery_lvl_val_hdl	Battery Level characteristic value handle
		battery_lvl_cfg_hdl	Battery Level characteristic configuration descriptor handle
		battery_lvl_prop	Battery Level property

3.3.13 RBLE_CPP_EVENT_COLLECTOR_DISABLE_COMP

RBLE_CPP_EVENT_COLLECTOR_DISABLE_COMP	
This event reports the result of disabling the Collector role (RBLE_CPP_Collector_Disable).	
Parameters:	
status	Result of disabling the Collector role (See *2.2* and *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for rBLE status*.)
conhdl	Connection handle

3.3.14 RBLE_CPP_EVENT_COLLECTOR_ERROR_IND

RBLE_CPP_EVENT_COLLECTOR_ERROR_IND	
This event indicates an error code unique to the CPP Collector role.	
Parameters:	
status	Error code (See *2.2* and *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for rBLE status*.)
conhdl	Connection handle

3.3.15 RBLE_CPP_EVENT_COLLECTOR_MEASUREMENTS_NTF

RBLE_CPP_EVENT_COLLECTOR_MEASUREMENTS_NTF			
This event indicates the measured value sent from the Sensor.			
Parameters:			
	conhdl	Connection handle	
	measure_info	*flags*	Flag that defines whether there is a data field in the characteristic value or not
		instant_power	Instantaneous power [W]
		pedal_balance	Pedal power balance [%]
		accumulated_torque	Accumulated torque [1/32 Nm]
		wheel_revolutions	Cumulative wheel revolutions
		wheel_event	Last wheel event time [1/2048 second]
		crank_revolutions	Cumulative crank revolutions
		crank_event	Last crank event time [1/1024 second]
		max_force_magnitude	Maximum force magnitude [N]
		min_force_magnitude	Minimum force magnitude [N]
		max_torque_magnitude	Maximum torque magnitude [1/32 Nm]
		min_torque_magnitude	Minimum torque magnitude [1/32 Nm]
		max_angle	Maximum angle [degree]
		min_angle	Minimum angle [degree]
		top_dead_spot	Top dead spot angle [degree]
		bottom_dead_spot	Bottom dead spot angle [degree]
		accumulated_energy	Accumulated energy [kJ]

3.3.16 RBLE_CPP_EVENT_COLLECTOR_VECTOR_NTF

RBLE_CPP_EVENT_COLLECTOR_VECTOR_NTF			
This event indicates the measured value sent from the Sensor. Depending on the flags information, value of force [N] or torque [1/32 Nm] is in the magnitude[ ].			
Parameters:			
	conhdl	Connection handle	
	vector_info	*flags*	Flag that defines whether there is a data field in the characteristic value or not
		crank_revolutions	Cumulative crank revolutions
		crank_event	Last crank event time
		first_crank_angle	First crank measurement angle
		array_num	Number of magnitude information
		magnitude[ ]	Instantaneous Force Magnitude Array [N]or Instantaneous Torque Magnitude Array [1/32 Nm]

3.3.17 RBLE_CPP_EVENT_COLLECTOR_CP_CP_IND

RBLE_CPP_EVENT_COLLECTOR_CP_CP_IND			
This event indicates the measurement interval sent from the Sensor. Check the request_op_code and the response_value whether the operation were sent by the CP control point characteristic setting (RBLE_CPP_Collector_Write_CP_Control_Point). If the response_value is RBLE_CPP_RES_SUCCESS_CODE, the value of the member corresponding to request_opcode is enabled. If the response_value is RBLE_CPP_OP_REQ_SUPPORTED_SL_CODE, the location_num and the supported_location are enabled. In the supported_location array, the number of information that is specified in the location_num is valid.			
Parameters:			
conhdl	Connection handle		
ind_cp_info	*OpCode*	RBLE_CPP_OP_RESPONSE_CODE	Response code
	request_op_code	RBLE_CPP_OP_SET_CUMULATIVE_CODE	Set cumulative value
		RBLE_CPP_OP_UPDATE_SL_CODE	Update sensor location
		RBLE_CPP_OP_REQ_SUPPORTED_SL_CODE	Request supported sensor locations
		RBLE_CPP_OP_SET_CRANK_LEN_CODE	Set crank length
		RBLE_CPP_OP_REQ_CRANK_LEN_CODE	Request crank length (set the crank_length)
		RBLE_CPP_OP_SET_CHAIN_LEN_CODE	Set chain length
		RBLE_CPP_OP_REQ_CHAIN_LEN_CODE	Request chain length (set the chain_length)
		RBLE_CPP_OP_SET_CHAIN_WEI_CODE	Set chain weight
		RBLE_CPP_OP_REQ_CHAIN_WEI_CODE	Request chain weight (set the chain_weight)
		RBLE_CPP_OP_SET_SPAN_LEN_CODE	Set span length
		RBLE_CPP_OP_REQ_SPAN_LEN_CODE	Request span length (set the span_length)
		RBLE_CPP_OP_START_OFFSET_COMPENSATION_CODE	Start offset compensation (set the offset_compensation)
		RBLE_CPP_OP_MASK_CP_MEAS_CONTENT_CODE	Mask cycling power measurement characteristic content
		RBLE_CPP_OP_REQ_SAMPL_RATE_CODE	Request sampling rate (set the sampling_rate)
		RBLE_CPP_OP_REQ_FACTORY_CALIB_DATE_CODE	Request factory calibration date (set the stamp)
	reqponse_value	RBLE_CPP_RES_SUCCESS_CODE	*Success*
		RBLE_CPP_RES_NOT_SUPPORTED_CODE	*Op Code not supported*

RBLE_CPP_EVENT_COLLECTOR_CP_CP_IND				
			RBLE_CPP_RES_INVALID_PARAM_CODE	*Invalid parameter*
			RBLE_CPP_RES_OP_FAILED_CODE	*Operation failed*
		crank_length	Crank length value	
		chain_length	Chain length value	
		chain_weight	Chain weight value	
		span_length	Span length value	
		offset_compensation	Offset compensation	
		sampling_rate	Sampling rate	
		stamp	*year*	Year
			month	Month
			day	Day
			hour	Hour
			min	Minute
			sec	Second
	location_num	Number of stored sensor location information		
	supported_location[RBLE_CPP_SENSORE_LOCATION_MAX]	RBLE_CPPC_SENSOR_OTHER		Other
		RBLE_CPPC_SENSOR_TOP_OF_SHOE		Top of shoe
		RBLE_CPPC_SENSOR_IN_SHOE		In shoe
		RBLE_CPPC_SENSOR_HIP		Hip
		RBLE_CPPC_SENSOR_FRONT_WHEEL		Front wheel
		RBLE_CPPC_SENSOR_LEFT_CRANK		Left crank
		RBLE_CPPC_SENSOR_RIGHT_CRANK		Right crank
		RBLE_CPPC_SENSOR_LEFT_PEDAL		Left pedal
		RBLE_CPPC_SENSOR_RIGHT_PEDAL		Right pedal
		RBLE_CPPC_SENSOR_FRONT_HUB		Front hub
		RBLE_CPPC_SENSOR_REAR_DROPOUT		Rear dropout
		RBLE_CPPC_SENSOR_CHAINSTAY		Chainstay
		RBLE_CPPC_SENSOR_REAR_WHEEL		Rear wheel
		RBLE_CPPC_SENSOR_REAR_HUB		Rear hub
		RBLE_CPPC_SENSOR_CHEST		Chest

3.3.18 RBLE_CPP_EVENT_COLLECTOR_READ_CHAR_RESPONSE

RBLE_CPP_EVENT_COLLECTOR_READ_CHAR_RESPONSE			
This event reports the response to the characteristic value read request (RBLE_CPP_Collector_Read_Char). Read out the read data in accordance with the contents of the request.			
Parameters:			
conhdl	Connection handle		
att_code	0x00	Characteristic value successfully acquired	
	Other than 0x00	Error occurred when acquiring characteristic value (See *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for ATT error code*.)	
data	*each_len*	Length of each result	
	len	Data length	
	data[RBLE_ATTM_MAX_VALUE]	Read characteristic data	

3.3.19 RBLE_CPP_EVENT_COLLECTOR_WRITE_CHAR_RESPONSE

RBLE_CPP_EVENT_COLLECTOR_WRITE_CHAR_RESPONSE		
This event reports the response to the characteristic value write request (RBLE_CPP_Collector_Write_Char).		
Parameters:		
conhdl	Connection handle	
att_code	0x00	Characteristic value successfully written
	Other than 0x00	Error occurred when writing characteristic value (See *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for ATT error code*.)

3.3.20 RBLE_CPP_EVENT_COLLECTOR_COMMAND_DISALLOWED_IND

RBLE_CPP_EVENT_COLLECTOR_COMMAND_DISALLOWED_IND		
This event indicates the error that occurs when a command executed by the Collector role cannot be accepted.		
Parameters:		
status	Result of command execution (See *2.2* and *Bluetooth Low Energy Protocol Stack API Reference Manual: Basics, 3.2, Declaration of enumerated type for rBLE status*.)	
opcode	RBLE_CMD_CPP_COLLECTOR_ENABLE	Collector role enable command
	RBLE_CMD_CPP_COLLECTOR_DISABLE	Collector role disable command
	RBLE_CMD_CPP_COLLECTOR_READ_CHAR	Characteristic read command
	RBLE_CMD_CPP_COLLECTOR_WRITE_CHAR	Characteristic write command
	RBLE_CMD_CPP_COLLECTOR_WRITE_CONTROL_POINT	CP control point write command

3.4 Message Sequence Chart

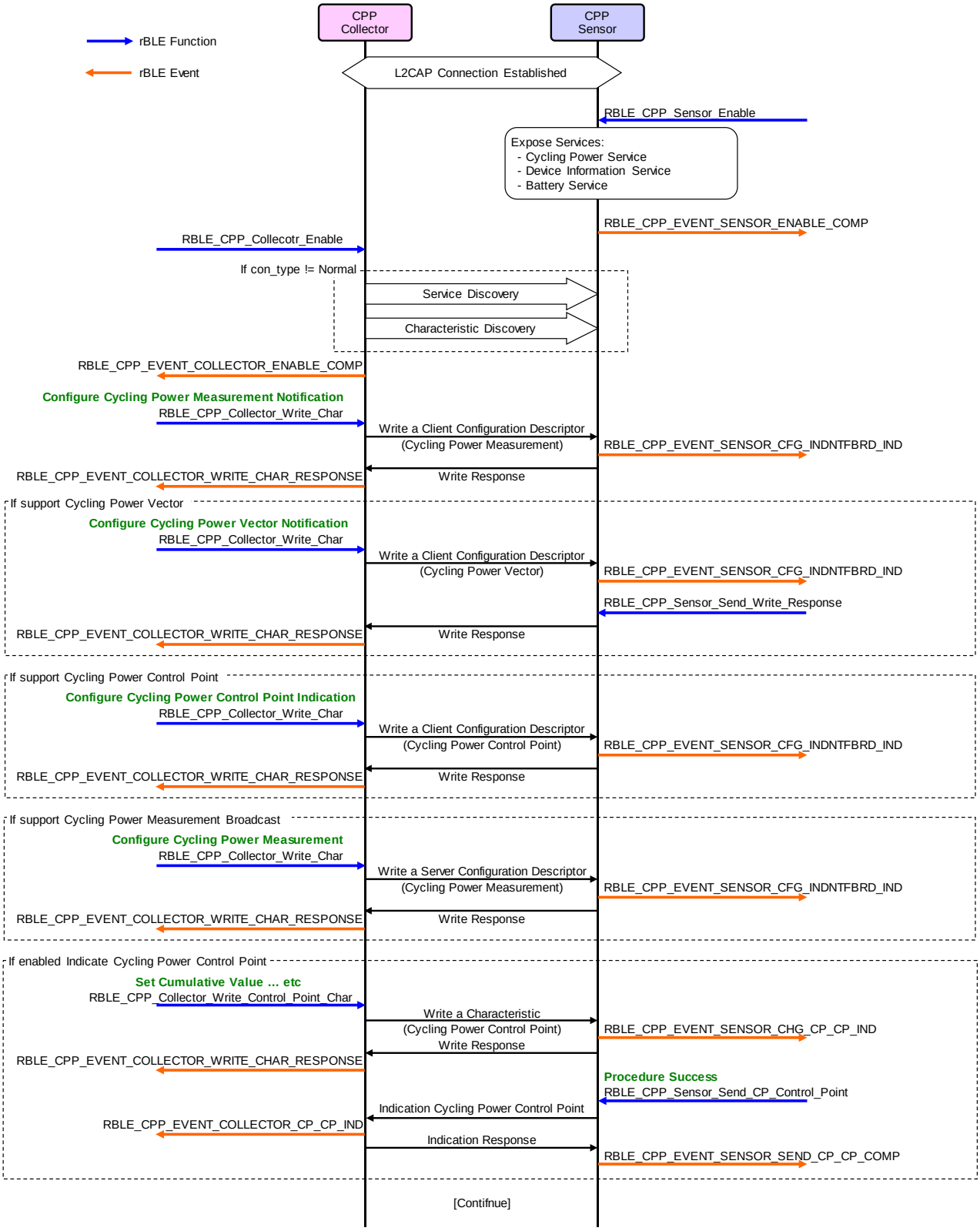

図 3-1 example of use case realization of CPP by using rBLE API

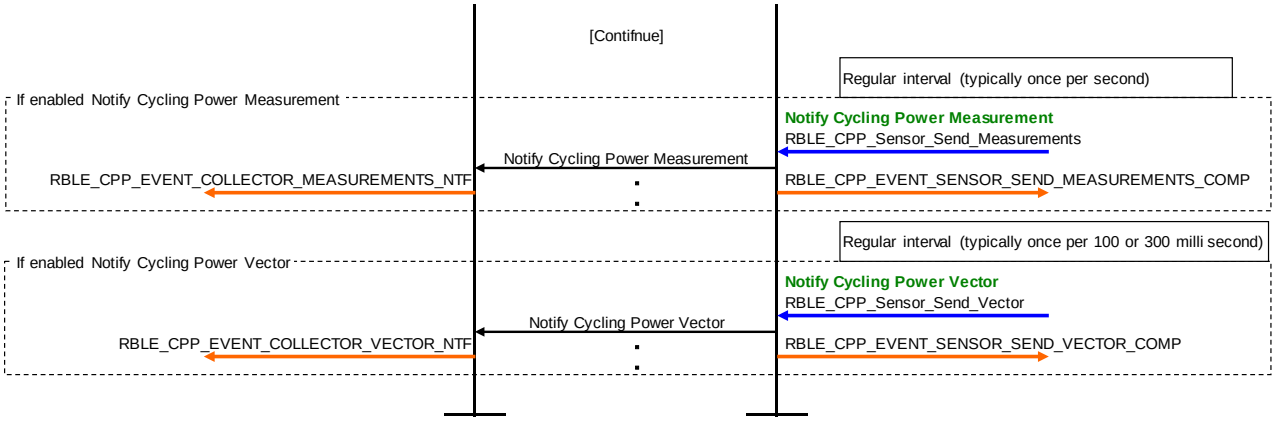

図 3-2 example of use case realization of CPP by using rBLE API

4. Notes

Broadcast function in Cycling Power Profile is not supported.

Therefore, cycling power measurement information cannot be broadcasted by using RBLE_CPP_Sensor_Broadcast_Measurements API.

Appendix A How to Read Definition Tables

This section shows how to read the tables that describes the rBLE API functions and events shown in this document.

A.1 How to Read Function Definition Tables

The following contents are included in the function definition tables:

The Parameters area describes the parameters specified for the function.
The italicized character strings on the left are the parameters of the function.
The meaning of each parameter is described on the far right following the variables.

The italicized character string(s) next to each parameter indicate the member(s) of the parameter (structure).

The values that can be specified for the parameter might be described between the parameter name and its description.

The function definition is shown at the top of the table in the row with the light green background. This area shows the function prototype.				
The operation of the function and the event reported after executing the function are described in this area.				
Parameters:				
	Parameter 1	Description of parameter 1		
	Parameter 2	*Member 1*	Value 1 that can be specified for member 1	Description of value 1 that can be specified for member 1
			Value 1 that can be specified for member 2	Description of value 1 that can be specified for member 2
		Member 2	Description of member 2	
Return:				
	Value 1 that might be returned	Description of value 1 that might be returned		
	Value 2 that might be returned	Description of value 2 that might be returned		

The Return area describes the values returned for the function.
The leftmost row shows the value that might be returned, and the next row describes the return value.

A.2 How to Read Event Definition Tables

The following contents are included in the event definition tables:

The Parameters area describes the parameters specified for the event.
The italicized character strings on the left show the parameters of the event parameter structure. The meaning of each parameter is described on the far right.

The italicized character string(s) next to each parameter indicate the member(s) of the parameter (structure).

The event definition is shown at the top of the table in the row with the orange background. This area shows the event type.			
The information reported by the event is described in this area.			
Parameters:			
	Parameter 1	Description of parameter 1	
	Parameter 2	*Member 1*	Description of member 1
		Member 2	Description of member 2
		Member 3	Description of member 3
	Parameter 3	Value 1 that can be specified for parameter 3	Description of value 1 that can be specified for parameter 3
		Value 2 that can be specified for parameter 3	Description of value 2 that can be specified for parameter 3

The values that can be specified for the parameter might be shown between the parameter name and its description.

Appendix B Referenced Documents

1. Bluetooth Core Specification v4.0, Bluetooth SIG
2. Find Me Profile Specification v1.0, Bluetooth SIG
3. Immediate Alert Service Specification v1.0, Bluetooth SIG
4. Proximity Profile Specification v1.0, Bluetooth SIG
5. Link Loss Service Specification v1.0, Bluetooth SIG
6. Tx Power Service Specification v1.0, Bluetooth SIG
7. Health Thermometer Profile Specification v1.0, Bluetooth SIG
8. Health Thermometer Service Specification v1.0, Bluetooth SIG
9. Device Information Service Specification v1.1, Bluetooth SIG
10. Blood Pressure Profile Specification v1.0, Bluetooth SIG
11. Blood Pressure Service Specification v1.0, Bluetooth SIG
12. HID over GATT Profile Specification v1.0, Bluetooth SIG
13. HID Service Specification v1.0, Bluetooth SIG
14. Battery Service Specification v1.0, Bluetooth SIG
15. Scan Parameters Profile Specification v1.0, Bluetooth SIG
16. Scan Parameters Service Specification v1.0, Bluetooth SIG
17. Heart Rate Profile Specification v1.0, Bluetooth SIG
18. Heart Rate Service Specification v1.0, Bluetooth SIG
19. Cycling Speed and Cadence Profile Specification v1.0, Bluetooth SIG
20. Cycling Speed and Cadence Service Specification v1.0, Bluetooth SIG
21. Cycling Power Profile Specification v0.9, Bluetooth SIG
22. Cycling Power Service Specification v0.9, Bluetooth SIG
23. Glucose Profile Specification v1.0, Bluetooth SIG
24. Glucose Service Specification v1.0, Bluetooth SIG
25. Bluetooth SIG Assigned Numbers https://www.bluetooth.org/Technical/AssignedNumbers/home.htm
26. Services & Characteristics UUID http://developer.bluetooth.org/gatt/Pages/default.aspx
27. Personal Health Devices Transcoding White Paper v1.2, Bluetooth SIG

Appendix C Terminology

Term	Description
Service	A service is provided from a GATT server to a GATT client. The GATT server exposes some characteristics as the interface. The service prescribes how to access the exposed characteristics.
Profile	A profile enables implementation of a use case by using one or more services. The services used are defined in the specifications of each profile.
Characteristic	A characteristic is a value used to identify services. The characteristics to be exposed and their formats are defined by each service.
Role	Each device takes the role prescribed by the profile or service in order to implement the specified use case.
Client Characteristic Configuration Descriptor	A descriptor is used to control notifications or indications of characteristic values that include the client characteristic configuration descriptor sent from the GATT server.
Server Characteristic Configuration Descriptor	A descriptor is used to control broadcast of characteristic values that include the server characteristic configuration descriptor sent from the GATT server.
Connection Handle	The handle determined by the controller stack and is used to identify connection with a remote device. The valid handle range is between 0x0000 and 0x0EFF.

REVISION HISTORY	Bluetooth Low Energy Protocol Stack API Reference Manual: CPP

Rev.	Date	Description	
		Page	Summary
0.10	Apr 5, 2013	---	Provisional Edition issued
0.11	Apr 12, 2013	---	Bookmark is added.
0.12	Sep 6, 2013	---	Definition of RBLE_BAS_CONTENT structure is changed to RBLE_BATS_CONTENT structure. Description of 3.3.10 is added comment. Parameters of 3.3.12 is changed to correct members. Parameters description of 3.3.18 and 3.3.19 is added comment. RBLE_CPP_Sensor_Send_Write_Response function is added. Correction of errors.
1.00	Nov 29, 2013	--- 24 44	First Edition issued Description of 3.2.8 is added comment. 3.4.Message Sequence Chart is added.
1.01	Sep 19, 2014	2 5 ---	The common definitions of profile are added. Definitions of client/server configuration characteristic value and connection type are deleted. Parameter description is changed to use the common definitions of profile.
1.02	Apr 17, 2015	2	The service definitions are updated.

Bluetooth Low Energy Protocol Stack
API Reference Manual: CPP

Publication Date: Rev.1.02 Apr 17, 2015

Published by: Renesas Electronics Corporation

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0

Bluetooth Low Energy Protocol Stack

Renesas Electronics Corporation

R01UW0099EJ0102