

BLE Glucose Meter

1.0

Features

- BLE GLS Service GATT Server role operation
- DeepSleep mode support
- Reporting the workflow status through UART
- LED status indication

General Description

This example project demonstrates the BLE Glucose Meter application workflow. The Glucose Meter application uses the BLE Glucose Profile to report glucose measurement records to the client. Also the Glucose Meter application uses the Battery Service to notify the battery level and the Device Information Service to assert the Device Name, etc.

Development Kit Configuration

Default CY8CKIT-042 BLE Pioneer Kit configuration,
Connect J2 pin P3[0] to J3 pin VREF.

Project Configuration

BLE Glucose Meter Example Project

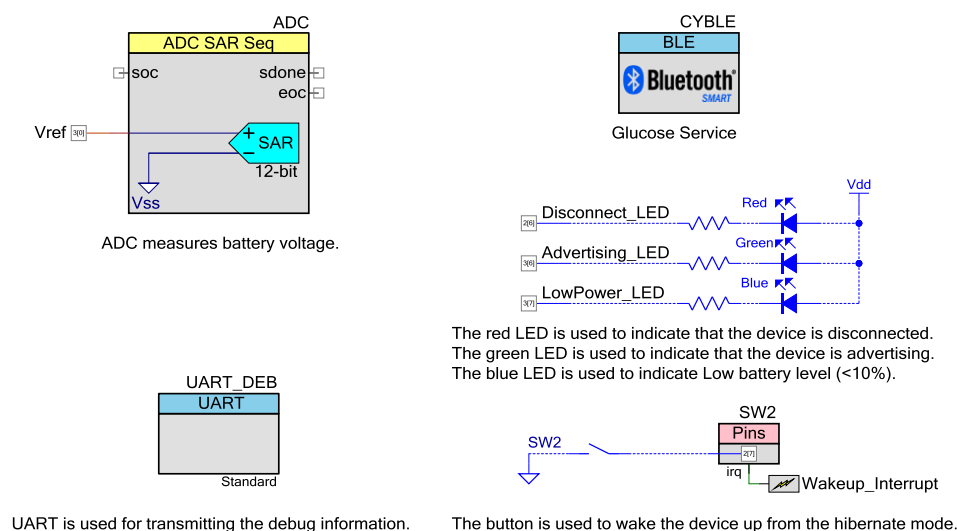


Figure 1. Top design schematic

The BLE component is configured as Glucose Server in the GAP Peripheral role. Also Battery and Device Information Services are included.

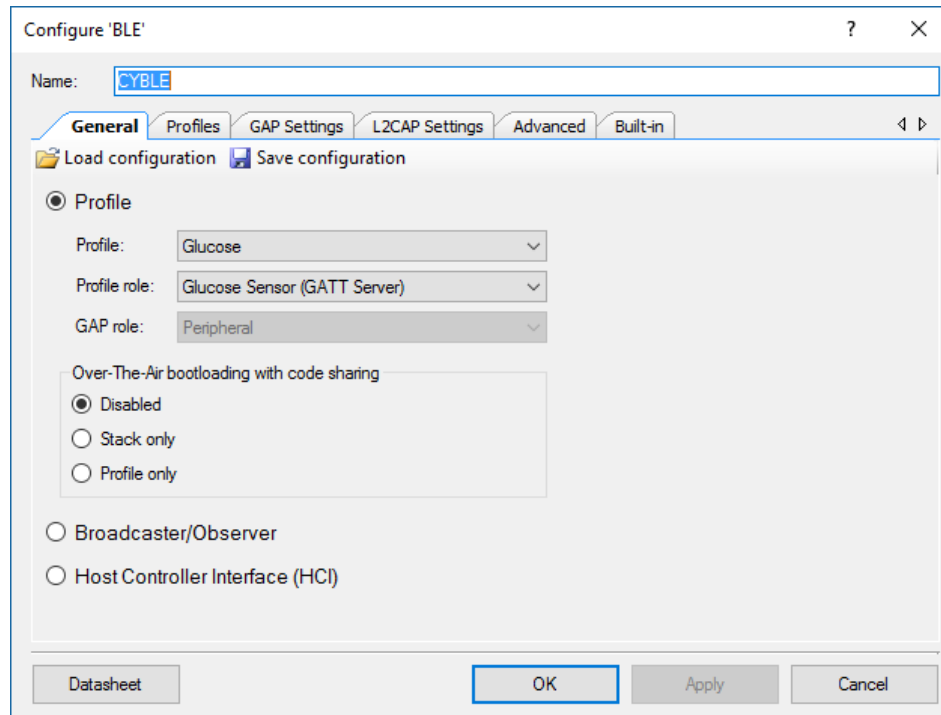


Figure 2. General settings

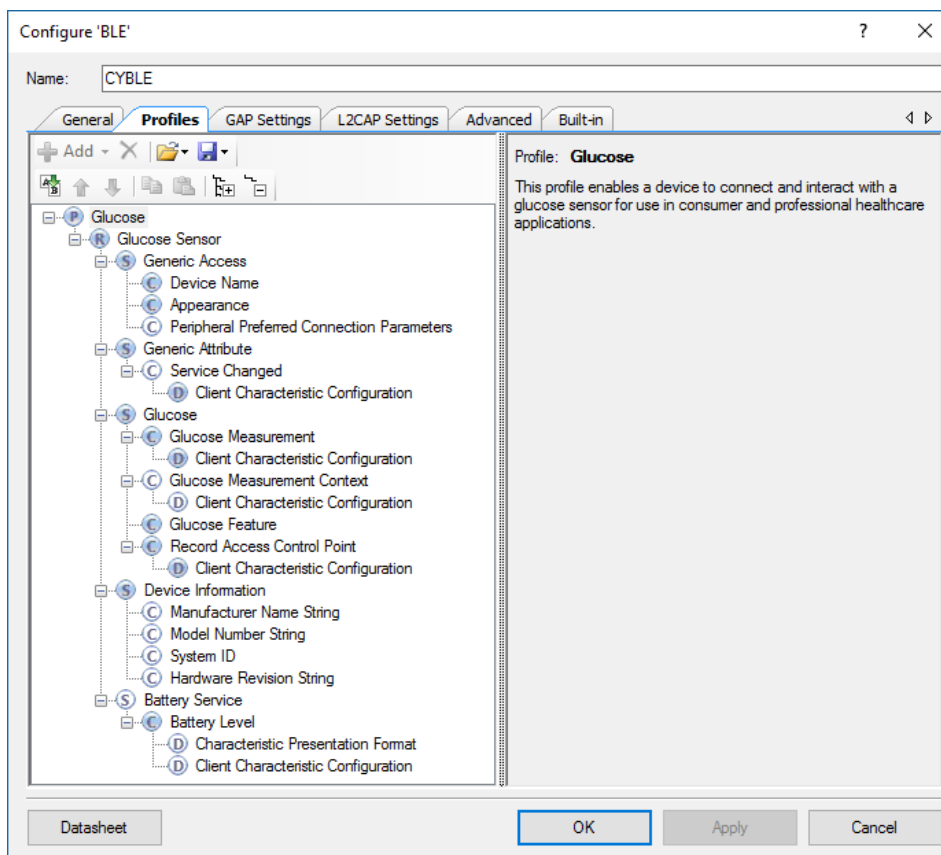


Figure 3. GATT settings

Configure 'BLE'

Name: CYBLE

General Profiles **GAP Settings** L2CAP Settings Advanced Built-in

General

- Peripheral role
 - Advertisement settings
 - Advertisement packet
 - Scan response packet
 - Peripheral preferred connection
 - Security

Restore Defaults

Device address

Public address (Company ID - Company assigned): 00A050-00000a

☐ Silicon generated "Company assigned" part of device address

You can use the user configuration section of the supervisory flash to store the public device address for mass production.

Device name: Glucose Meter

Appearance: Generic Glucose Meter

Attribute MTU size (bytes): 23

Link layer max TX payload size (bytes): 27

Link layer max RX payload size (bytes): 27

Adv/Scan TX power level (dBm): 0

Connection TX power level (dBm): 0

☐ Enable Link Layer Privacy

Datasheet OK Apply Cancel

Figure 4. GAP settings

Configure 'BLE'

Name: CYBLE

General Profiles **GAP Settings** L2CAP Settings Advanced Built-in

General

- Peripheral role
 - Advertisement settings**
 - Advertisement packet
 - Scan response packet
 - Peripheral preferred connection
 - Security

Restore Defaults

Discovery mode: General

Advertising type: Connectable undirected advertising

Filter policy: Scan request: Any | Connect request: Any

Advertising channel map: All channels

Advertising interval

Fast advertising interval:

Minimum (ms): 20

Maximum (ms): 30

☒ Timeout (s): 30

☒ Slow advertising interval:

Minimum (ms): 1000

Maximum (ms): 2500

☒ Timeout (s): 150

Datasheet OK Apply Cancel

Figure 5. GAP settings -> Advertisement settings

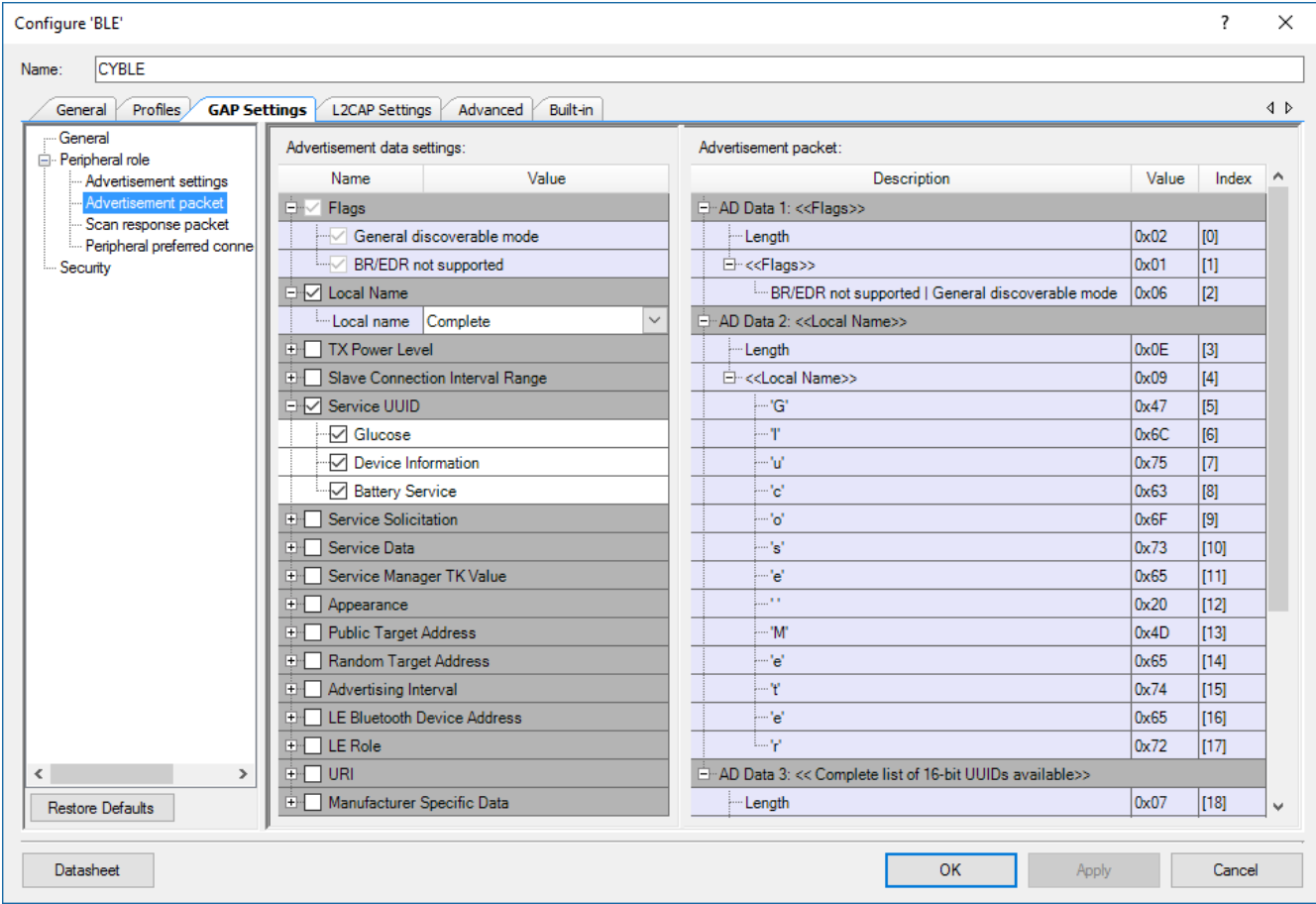


Figure 6. GAP Settings -> Advertisement packet

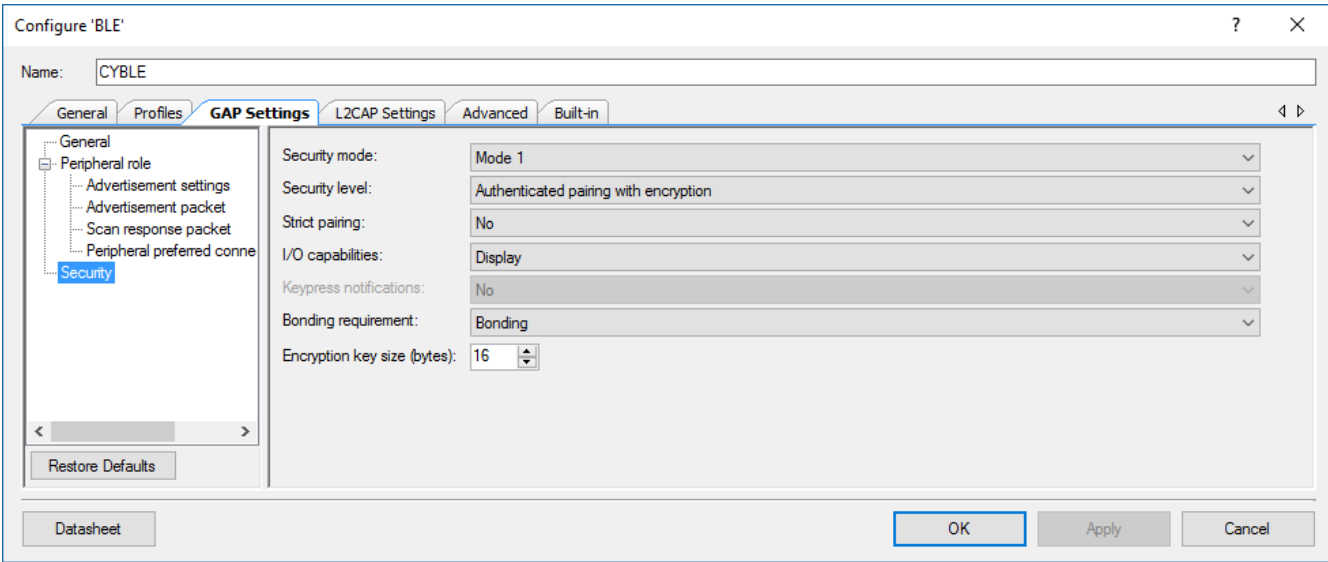


Figure 7. GAP Settings -> Security

Project Description

The project demonstrates the core functionality of the BLE component configured as a Glucose Server.

Right after startup the device performs BLE component initialization. In this project three callback functions are required for the BLE operation. One callback function (AppCallBack()) is required to receive generic events from BLE Stack and the service-specific callbacks BasCallBack() and GlSCallBack() for Battery and Glucose service-specific events accordingly. The CYBLE_EVT_STACK_ON event indicates a successful initialization of BLE Stack. After this event is received, the component starts advertising with the packet structure as configured in the BLE component customizer (see **Figure 6**). The BLE component stops advertising as soon as 180 seconds advertising period expires.

The Glucose Meter device can be connected to any BLE (4.0 or later) compatible device configured as GAP Central role and GATT Client which supports Glucose Profile. Also the Battery and Device Information Services may be optionally used. To connect to the Glucose Meter device, send a connection request to the device while the device is advertising. The green LED is blinking while the device is advertising. The red LED is turned on after disconnection to indicate that no Client is connected to the device. When the Client connects successfully, both red and green LEDs are turned off. The Glucose Meter device requires the authentication, the IO capability is “display only” (see Figure 7. GAP Settings -> Security) i.e. the Glucose Meter device indicates the passkey through UART and user should enter that passkey into Client device. If the Client is paired with the Glucose Meter, firstly the Glucose Measurement, Glucose Measurement Context (if it is supported by client) characteristic notifications and the Record Access Control Point (RACP) characteristic indication should be enabled. Then the RACP characteristic can be written to assert any Glucose RACP requests (for details, see the Glucose Profile and Glucose Service specifications adopted by Bluetooth SIG). When the RACP request is asserted, the Client should wait for any Glucose Measurement, Glucose Measurement Context (if it is supported by the Client) characteristic notifications and the Record Access Control Point (RACP) characteristic indication (dependent on asserted request), or write the “Abort Operation” command into RACP characteristic. The WDT is used to timing the simulations, measurements and LED blinking.

While connected to the Client and between connection intervals, the device is put into Sleep Mode.

The blue LED is used for indicating the low battery level (<10%). **Note** that after the first connection establishment and until either device is wen to sleep or resented the blue LED will continuously glowing indicating the low battery.

Expected Results

The project sends the Glucose Service characteristic's notifications/indications and Battery Level notifications to the Central Client device. LEDs are blinking as described in Project Description section. Also the project sends log messages through the UART.

The project is intended to work in pair with any BLE-compatible device (e.g. phone, tablet) with appropriate software (with e.g. Android, iOS with installed application which supports

Glucose Profile). You can use CySmart mobile app (available for [Android](#) and [iOS](#)) as the Glucose client:

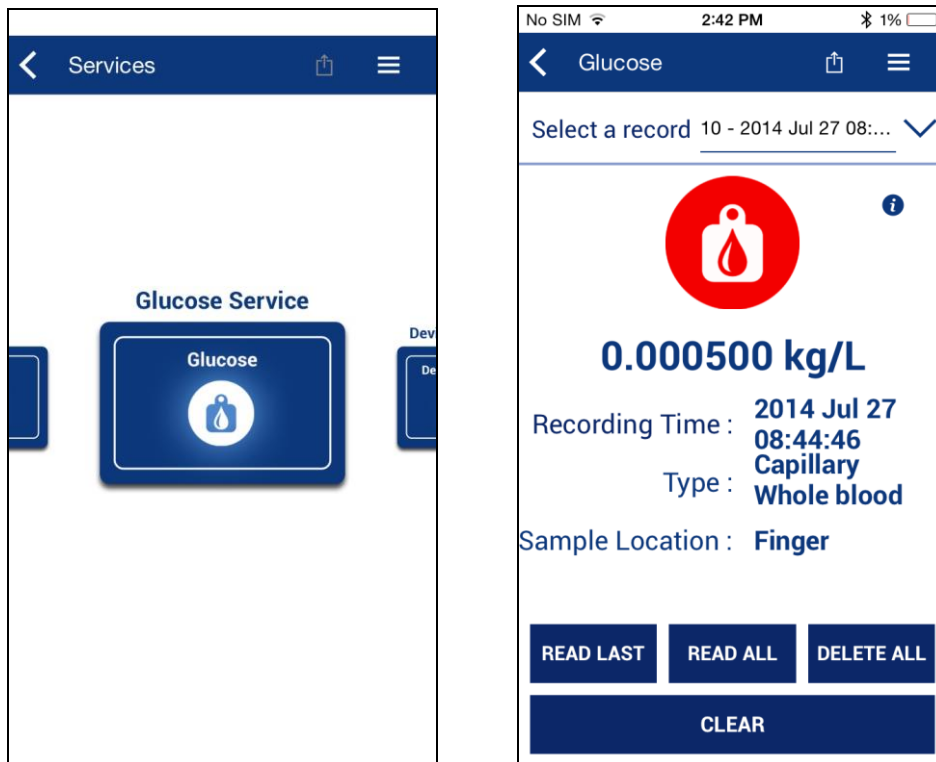


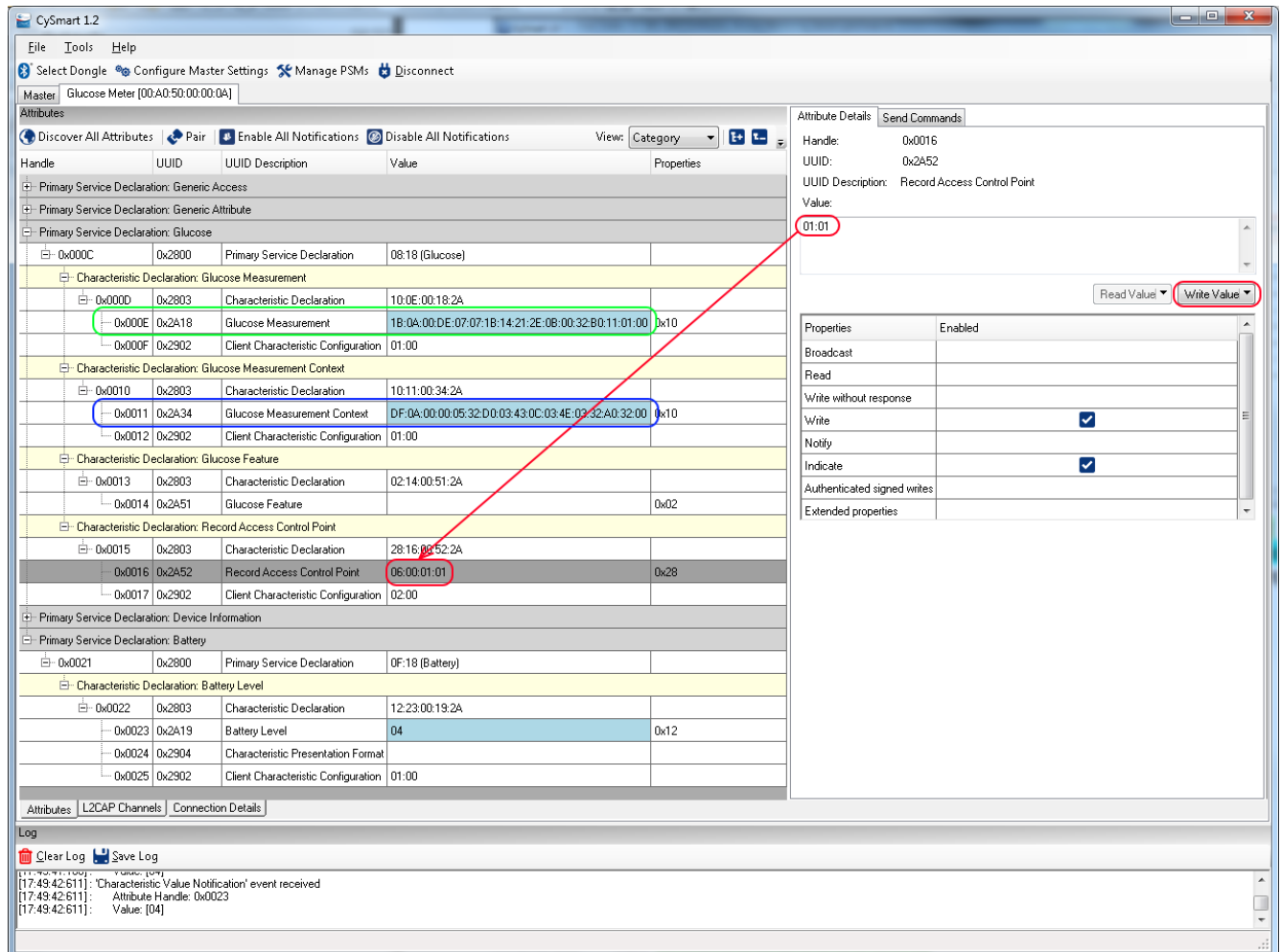
Figure 8. CySmart iOS app

Also, the Glucose Meter can be used together with [CySmart app for Windows](#). It is required to match the security settings between Glucose Meter and CySmart Client and perform pairing (bonding) before any writing (enabling notifications etc.) into Server's GATT database. For further instructions on how to use CySmart application, see [CySmart User Guide](#).

The simple example how to use CySmart Windows application as Glucose Service client is the next:

- Connect the CySmart BLE dongle to a USB port on the PC.
- Launch CySmart app and select connected dongle in the dialog window.
- Reset the development kit to start advertising by pressing SW1 button.
- Click **Start Scan** button to discover available devices.
- Select **Glucose Meter** in the list of available devices and connect to it.
- Click **Pair**, enter the 6-digit passkey which is displayed through the terminal, then **Discover All Attributes**, then **Read All Characteristics**, and finally **Enable All Notifications** in CySmart app.
- To get the Glucose Service functionality, for example, select the **Record Access Control Point (RACP)** characteristic value and write the command **"01:01"** which

means “Report All Glucose Measurement Records” (all these commands are described in detail in the [GLS Specification](#)):



- Observe the CySmart's log window: Server sends eleven **Glucose Measurement characteristic notifications** and three **Glucose Measurement Context characteristic notifications** and then the **RACP** indication **"06 00 01 01"** which means "The < Report All Glucose Measurement Records > command is performed successfully". An example of successful execution of "Report All Glucose Measurement Records" is shown on the following extract from the CySmart log:

```
[17:45:08:757]: 'Write Characteristic Value' request sent
[17:45:08:757]: Attribute Handle: 0x0016
[17:45:08:757]: Value: 01:01
[17:45:08:762]: 'Command Status' event received
[17:45:08:762]: Status: BLE_STATUS_OK
[17:45:08:777]: 'Command Complete' event received
[17:45:08:777]: Status: BLE_STATUS_OK
[17:45:08:805]: 'Characteristic Value Notification' event received
[17:45:08:805]: Attribute Handle: 0x000E
[17:45:08:805]: Value: 09:00:00:DE:07:07:1B:14:1E:28:00:00:01:00
[17:45:08:805]: 'Characteristic Value Notification' event received
[17:45:08:805]: Attribute Handle: 0x000E
[17:45:08:805]: Value: 17:01:00:DE:07:07:1B:14:1E:28:01:00:32:D0:11
[17:45:08:805]: 'Characteristic Value Notification' event received
[17:45:08:805]: Attribute Handle: 0x0011
[17:45:08:805]: Value: DF:01:00:00:05:32:D0:03:43:0C:03:4E:03:32:A0:32:00
[17:45:08:805]: 'Characteristic Value Notification' event received
[17:45:08:805]: Attribute Handle: 0x000E
[17:45:08:805]: Value: 0B:02:00:DE:07:07:1B:14:1E:28:02:00:32:B0:11:01:00
```

```
[17:45:08:805]: 'Characteristic Value Notification' event received
[17:45:08:805]:   Attribute Handle: 0x000E
[17:45:08:805]:   Value: 09:03:00:DE:07:07:1B:14:1E:28:3C:00:01:00
[17:45:08:805]: 'Characteristic Value Notification' event received
[17:45:08:805]:   Attribute Handle: 0x000E
[17:45:08:805]:   Value: 17:04:00:DE:07:07:1B:14:1E:28:3C:00:32:D0:11
[17:45:08:806]: 'Characteristic Value Notification' event received
[17:45:08:806]:   Attribute Handle: 0x0011
[17:45:08:806]:   Value: DF:04:00:00:05:32:D0:03:43:0C:03:4E:03:32:A0:32:00
[17:45:08:807]: 'Characteristic Value Notification' event received
[17:45:08:807]:   Attribute Handle: 0x000E
[17:45:08:807]:   Value: 0B:05:00:DE:07:07:1B:14:1E:28:3B:00:32:B0:11:01:00
[17:45:08:810]: 'Characteristic Value Notification' event received
[17:45:08:810]:   Attribute Handle: 0x000E
[17:45:08:810]:   Value: 09:06:00:DE:07:07:1B:14:1E:28:C4:FF:01:00
[17:45:08:814]: 'Characteristic Value Notification' event received
[17:45:08:814]:   Attribute Handle: 0x000E
[17:45:08:814]:   Value: 17:07:00:DE:07:07:1B:14:1E:28:C4:FF:32:D0:11
[17:45:08:817]: 'Characteristic Value Notification' event received
[17:45:08:817]:   Attribute Handle: 0x0011
[17:45:08:817]:   Value: DF:07:00:00:05:32:D0:03:43:0C:03:4E:03:32:A0:32:00
[17:45:08:819]: 'Characteristic Value Notification' event received
[17:45:08:819]:   Attribute Handle: 0x000E
[17:45:08:819]:   Value: 0B:08:00:DE:07:07:1B:14:1E:28:C6:FF:32:B0:11:01:00
[17:45:08:822]: 'Characteristic Value Notification' event received
[17:45:08:822]:   Attribute Handle: 0x000E
[17:45:08:822]:   Value: 0B:09:00:DE:07:07:1B:14:20:2D:0A:00:37:B0:11:01:00
[17:45:08:827]: 'Characteristic Value Notification' event received
[17:45:08:827]:   Attribute Handle: 0x000E
[17:45:08:827]:   Value: 1B:0A:00:DE:07:07:1B:14:21:2E:0B:00:32:B0:11:01:00
[17:45:08:827]: 'Characteristic Value Indication' event received
[17:45:08:827]:   Attribute Handle: 0x0016
[17:45:08:827]:   Value: 06:00:01:01
[17:45:08:871]: 'Characteristic Value Notification' event received
[17:45:08:871]:   Attribute Handle: 0x0011
[17:45:08:871]:   Value: DF:0A:00:00:05:32:D0:03:43:0C:03:4E:03:32:A0:32:00
[17:45:09:617]: 'Characteristic Value Notification' event received
[17:45:09:617]:   Attribute Handle: 0x0023
[17:45:09:617]:   Value: [04]
[17:45:11:113]: 'Characteristic Value Notification' event received
[17:45:11:113]:   Attribute Handle: 0x0023
[17:45:11:113]:   Value: [04]
```


The correspondent example UART log is shown below:

```
BLE Glucose Meter Example Project
EVT_STACK_ON
Start Advertisement with addr: 00a05000000a
CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP, state: 3
EVT_GATT_CONNECT_IND: attId 0, bdHandle 4
EVT_GAP_DEVICE_CONNECTED: 4
EVT_GATTS_XCNHG_MTU_REQ
EVT_GAP_AUTH_REQ
EVT_GAP_PASSKEY_DISPLAY_REQUEST: 045073
EVT_GAP_ENCRYPT_CHANGE: 1
EVT_GAP_AUTH_COMPLETE: security:2, bonding:1, ekeySize:10, authErr 0
Store bonding data, status: 28
Store bonding data, status: 0
RACP characteristic indication is enabled
Store bonding data, status: 0
Glucose Measurement Context characteristic notification is enabled
Store bonding data, status: 0
Glucose Measurement characteristic notification is enabled
Store bonding data, status: 0
BAS event: 11a, EVT_BASS_NOTIFICATION_ENABLED: 0
Store bonding data, status: 0
MeasureBatteryLevelUpdate: 0
MeasureBatteryLevelUpdate: 0
RACP is written: 01 01
Opcode: Report stored records
Operator: All records
Glucose Ntf: 0
Glucose Ntf: 1
Glucose Context Ntf: 1
Glucose Ntf: 2
Glucose Ntf: 3 |
Glucose Ntf: 4
Glucose Context Ntf: 4
Glucose Ntf: 5
Glucose Ntf: 6
Glucose Ntf: 7
Glucose Context Ntf: 7
Glucose Ntf: 8
Glucose Ntf: 9
Glucose Ntf: 10
Glucose Context Ntf: 10
RACP Ind: 6 0 1 1
RACP characteristic indication is confirmed
MeasureBatteryLevelUpdate: 0
MeasureBatteryLevelUpdate: 0
```

© Cypress Semiconductor Corporation, 2009-2016. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.