Delta mbed Platform

BLE Command Line Interface Document

Index:

Revision History

Version	Date	Reason of change	Maker
1.0	Jan 11, 2015	Initial release	Gill Wei
1.1	Mar 2, 2015	Add Wi-Fi Command Set	Marcus Chiou
1.2	Mar 10, 2015	Add BLE profile overview to introduction and BLE example for creating the sample by command	Gill Wei
1.3	Mar 27, 2015	Modify WIFI Command description	Marcus Chiou
1.4	Mar 30, 2015	Able to add BLE GATT 128bit vendor specific services/characteristics, modify advertise and add service command format.	Gill Wei
1.5	Jun 11, 2015	Modify 5.2 Procedure for creating an Profile about command description	Gill Wei
1.6	Sep 4, 2015	Merge mbed (mbed,BLE_API, RF51822) library, modify BLE name command, Wifi TCP/UDP blocking description	Gill Wei
1.7	Dec 9, 2015	Update GPIO PIN define to EVB version 3 Add BLE central mode API, WIFI device API, simplified command set.	Gill Wei
1.8	Jan 25, 2016	Add BLE data interrupt command; Chap 6.5 Wi-Fi data receive interrupt	Gill Wei
1.9	Jan 26, 2016	Remove Chap 6.5 Wi-Fi data receive interrupt, add Wifi data interrupt command; Modify Chap6 Wifi example program; Modify Chap5.3 BLE example procedure	Gill Wei
1.10	Feb 18, 2016	Modify BLE name, Wifi TCP send, UDP send format	Gill Wei
1.11	Feb 25, 2016	Modify TCP and UDP max data length	Gill Wei
1.12	April 25, 2016	Modify BLE Update command(add raw data type), Fix update	Gill Wei

		and readData command value length issue	
1.13	April 25, 2016	Fix Characteristic property setting issue,	Gill Wei
1.14	Sep 8, 2016	Add BLE Central mode commands	Silvia Chen
1.15	Sep 21, 2016	Fix BLE connect command issue; Modify the document for whole BLE module	Silvia Chen
1.16	Oct 20, 2016	Update Section 3.1 and Section 3.2	Tsungta Wu

Bluetooth Low Energy Module
Command Line Interface

A Command Line Interface (CLI) used to implement function of BLE Module.

1. Introduction

This document describes the function of each command within Delta mbed platform command line interface; include device configuration, BLE connection, sleep mode, and so on. This document also makes a demonstration of how to set up environment in PC and create a BLE profile in GATT server; below we use Glucose Profile setting as example, and list the typical procedure of standard profile setting.

Glucose profile is one of the officially defined BLE profile by Bluetooth official group 「SIG」, which is abbreviated of Special Interest Group. See Reference 1.

Profile defines and explain stored data format, one profile contains one or several services, and each service contains data for communication.

Glucose profile composed of Glucose Service and Device Information Service(see Figure.1), which with specific data format, for example, the Glucose Service contains data as Time, Sequence number, Glucose concentration or other data related to Glucose measurement.

In this case, Delta platform with Glucose sensor can be seem as 「Glucose Sensor」 (see Figure.1), the Central Device, like smart phone or laptop, can act as 「Glucose Collector」, which collect and display the glucose data transmitted from glucose sensor. Collector and sensor can also be treated as 「Central」 and 「Peripheral」, Reference 2 have BLE GATT Service overview and have explain more detail.

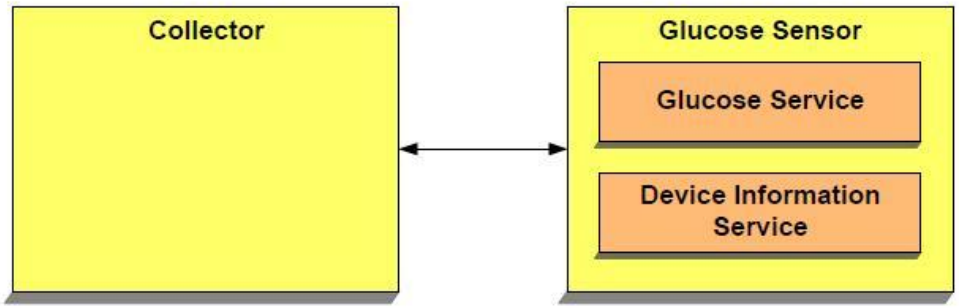

Figure.1 Glucose role and services

After define all the data in Glucose sensor, let us see how BLE device set up the connection and transmit data, Figure.2 show how BLE device in and out from different state; only in connecting state, sensor data will transfer in profile-defined format.

Supported Platform

- DFCM-NNN40-EVB
- DFBM-NQ620-EVB

Reference

1. Glucose Profile
https://developer.bluetooth.org/gatt/services/Pages/ServiceViewer.aspx?u=org.bluetooth.service.glucose.xml

2. SIG BLE GATT Service List
https://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx

2. Commands Status Responses

Whenever a command is received by device, an "OK" or "ERROR" response will be send back to host side. ERROR case will output with specific error string; OK will output with additional response for some specific commands.

2.1 OK

OK

Function: All successful commands will respond with OK message

Example:

RESPONSE: OK<cr_lf>
<cr_lf>

2.2 ERROR

ERROR

Function: All failure commands will respond with ERROR response with mapping error message

Command Format: ERROR;<error message>;

Response Values:

Error Message:

"No such command;",

"Wrong number of arguments;",

"Argument out of range;",

"Argument syntax error;",

"No matched argument;",

"Wrong command order;",

"Invalid state to perform operation;",

"Function call fail;"

Example:

RESPONSE: ERROR; Argument out of range;<cr_lf>
<cr_lf>

3. BLE Commands Description

This section describes the detail function of each command, including command format, argument format, and command description.

3.1 GPIO

CONTROL GPIO PIN

Function: Set or Clear a specific GPIO pin on BLE module

Command Format: cynb gpio <LED-NO> <SET/CLEAR>

Example:

COMMAND: cynb gpio 1 set<cr>
RESPONSE: <cr_lf>
OK<cr_lf>
<cr_lf>

Note: Available <LED-NO> number are 1, 2, 3 and 4 which are corresponded with LED1, LED2, LED3 and LED4 on the EVB respectively.

3.2 System Off

SLEEP MODE

Function: Go to system off mode, and then choose a specific pin; detect pin state change for wake up.

Command Format: cynb sleep <BUTTON-NO>

Example:

COMMAND: cynb sleep 1<cr>
RESPONSE: <cr_lf>
OK<cr_lf>
<cr_lf>

Note: Available <BUTTON-NO> number are 1, 2, 3 and 4 which are corresponded with BUTTON1, BUTTON 2, BUTTON 3 and BUTTON 4 on the EVB respectively. For NNN40 platform only BUTTON1 can be configured in this function.

3.3 Reset

RESET

Function: Reset BLE module.

Command Format: cynb reset

Example:

COMMAND: cynb reset<cr>
RESPONSE: <cr_lf>
OK<cr_lf>
<cr_lf>

3.4 System Information

MODULE INFORMATION

Function: List system information stored in module, which included Firmware version and Module name

Command Format: cynb info

Example:

COMMAND: cynb info<cr>
RESPONSE: <cr_lf>
OK; DELTA_CLI_V1.7;DFCM-NNN40-DT0R; <cr_lf>
<cr_lf>

3.5 Device Name

SET BLE DEVICE NAME

Function: Set friendly name for BLE module.

Command Format: cynb name <LENGTH> <NAME>

Example:

COMMAND: cynb name 9 DELTA CLI<cr>
RESPONSE: <cr_lf>
OK;DELTA CLI <cr_lf>
<cr_lf>

Note: Please type string length including “space”. Whenever initialize BLE module, default module name is “nRF5x”.

GET BLE DEVICE NAME

Function: Get the current device name for BLE module.

Command Format: cynb name

Example:

COMMAND: cynb name<cr>
RESPONSE: <cr_lf>
OK; nRF5x;<cr_lf>
<cr_lf>

Note: Default name is “nRF5x”

3.6 Init BLE Central Mode

INIT BLE CENTRAL STACK

Function: Init BLE Central stack.

Command Format: cynb initBleCen

Example:

COMMAND: cynb initBleCen<cr>
RESPONSE: <cr_lf>
OK<cr_lf>
<cr_lf>

3.7 BLE Start Advertising

START DEVICE ADVERTISING

Function: Start BLE advertising with specific timeout and interval.

Command Format: cynb advStart <INTERVAL> <TIMEOUT>

Example:

COMMAND: cynb advStart 64 180<cr>
RESPONSE: <cr_lf>
OK<cr_lf>
<cr_lf>

Note: If the system is already in advertising state, error message show that the system in invalid state for operation. The unit of INTERVAL is minisecond; The unit of TIMEOUT is second, 180 means that BLE advertising will stop after 180 seconds.
Default interval value is 64 ms, default time-out value is 180 seconds.
INTERVAL range: Maximum: 10240 Minimum:20
TIMEOUT range: Maximum: 16383 Minimum:1

3.8 BLE Stop Advertising

STOP DEVICE ADVERTISING

Function: Stop BLE advertising.

Command Format: cynb advStop

Example:

COMMAND: cynb advStop<cr>
RESPONSE: <cr_lf>
OK<cr_lf>
<cr_lf>

3.9 TX Power

RF TX POWER

Function: Set RF TX power for BLE module.

Command Format: cynb txPow <TX POWER>

Example:

COMMAND: cynb txPow 0<cr>
RESPONSE: <cr_lf>
OK;<cr_lf>
<cr_lf>

- Available TX power for NNN40 are -30, -20, -16, -12, -8, -4, 0, 4. (unit: dBm)
- Available TX power for NQ620 are -20 to +4 dBm in 4 dB steps

3.10 BLE Address

SET BLE Address

Function: Set BLE MAC address at run time.

Command Format: cynb bleAddr <BLE ADDR>

Example:

COMMAND: cynb bleAddr 0xE6BCA12B322F<cr>
RESPONSE: <cr_lf>
OK<cr_lf>
<cr_lf>

Note: BLE MAC address should be 12 hex numbers.

GET BLE Address

Function: Get current BLE MAC address.

Command Format: cynb bleAddr

Example:

COMMAND: cynb bleAddr<cr>
RESPONSE: <cr_lf>
OK;[E6 BC A1 2B 32 2F];<cr_lf>
<cr_lf>

3.11 BLE GATT Service

GATT SERVICE SETTING

Function: Add SIG defined or Vendor Specific service to server

Command Format: cynb gattService <SERVICE UUID>

Example:

COMMAND: cynb gattService 0x180A<cr>
RESPONSE: <cr_lf>
OK<cr_lf>
<cr_lf>

Note: Once one service had added on GATT server, characteristics can be added on, but **once registered new service, user can not add new characteristic in previous set service**. Service UUID can be 4 or 32 hex numbers.

GATT CHARACTERISTIC SETTING

Function: Add new characteristic on currently add services, including set characteristic UUID, characteristic property and value for BLE module.

Command Format: cynb gattChar <CHAR UUID> <ATTR PROP> <ATTR VALUE>

Example:

COMMAND: cynb gattChar 0x2A19 0xFF 0x1234<cr>
RESPONSE: <cr_lf>
OK<cr_lf>
<cr_lf>

Note: The characteristic property currently support multiple properties, including

0x00: NONE
0x01: BROADCAST
0x02: READ
0x04: WRITE_WITHOUT_RESPONSE
0x08: WRITE
0x10: NOTIFY
0x20: INDICATE
0x40: AUTHENTICATED_SIGNED_WRITES
0x80: EXTENDED_PROPERTIES

Select the appropriate and mandatory properties to specific characteristic, using bit mask, for example, if user want to add one characteristic with Notify and Read property, filled with 0x12.

- The total value field length is 10 bytes, equal to 20 hex numbers.
- Attribute value type is uint8_t, so the input hex number must be even number, ex. 0, 2, 4, etc.
- Characteristic UUID can be 4 or 32 hex numbers.
- Cannot add characteristic before not add any service, or error message will show: “Invalid state to perform operation.”

REGISTER GATT SERVICE

Function: Register SIG defined or Vendor Specific service to server
Command Format: cynb regService

Example:
COMMAND: cynb regService<cr>
RESPONSE: <cr_lf>
OK<cr_lf>

3.12 GATT Write Characteristic Value

UPDATE CHARACTERISTIC VALUE (BLE PERIPHERAL MODE)

Function: Change characteristic value and make an indication or notification according to the property of characteristic.

Command Format: cynb update <SERVICE UUID> <CHAR UUID> <TYPE> <VALUE>

Example (1):
COMMAND: cynb update 0x180D 0x2A39 0 1234<cr>
RESPONSE: <cr_lf>
OK; 31323334<cr_lf>
<cr_lf>
RESULT: This example update data to 0x31323334, data length have 4 bytes

Example (2):
COMMAND: cynb update 0x180D 0x2A39 1 0x1234<cr>
RESPONSE: <cr_lf>
OK;1234<cr_lf>
<cr_lf>
RESULT: This example update data to 0x1234, data length have 2 bytes

Note:
<TYPE> parameter have following values:

0 : Raw Data (Not transferred from ASCII code)
1 : Char Data

If the specified characteristic property contains notify/indicate, data update require BLE Central (ex. mobile phone) enable the notify/indicate descriptor (CCCD) previously.

- Attribute value type is uint8_t, so the input hex number must be even number, ex. 2, 4, 6, etc. Maximum data length is 20 bytes.

WRITE CHARACTERITSTIC VALUE (BLE CENTRAL MODE)

Function: BLE Central write value to specific characteristics.

Command Format: cynb cenWriteData <SERVICE UUID> <CHAR UUID> <TYPE> <VALUE>

Example (1):

COMMAND: cynb cenWriteData 0x180D 0x2A39 0 1234<cr>
RESPONSE: <cr_lf>
OK; 31323334<cr_lf>
<cr_lf>

Example (2):

COMMAND: cynb cenWriteData 0x180D 0x2A39 1 0x1234<cr>
RESPONSE: <cr_lf>
OK; 1234<cr_lf>
<cr_lf>

Note:

<TYPE> parameter have following values:
0 : Raw Data (Not transferred from ASCII code)
1 : Char Data

Attribute value type is uint8_t, so the input hex number must be even number, ex. 2, 4, 6, etc. Maximum data length is 20 bytes.

3.13 GATT Read Characteristic Value

READ CHARACTERITSTIC VALUE (BLE PERIPHERAL MODE)

Function: Read current characteristic value in BLE module.

Command Format: cynb readData <SERVICE UUID> <CHAR UUID>

Example:

COMMAND: cynb readData 0x180D 0x2A39<cr>
RESPONSE: <cr_lf>
OK;0x1234;<cr_lf>
<cr_lf>

Note: Max value field contains with 20 bytes, which equal to 40 hex numbers.

READ CHARACTERITSTIC VALUE (BLE CENTRAL MODE)

Function: BLE Central read value of specific characteristics.

Command Format: cynb cenReadData <SERVICE UUID> <CHAR UUID>

Example:

COMMAND: cynb cenReadData 0x180D 0x2A39<cr>
RESPONSE: <cr_lf>
OK;0x1234;<cr_lf>
<cr_lf>

Note: Max value field contains with 20 bytes, which equal to 40 hex numbers.

3.14 GATT Notification

ENABLE NOTIFICATION

Function: BLE Central enable notification of specific characteristics.

Command Format: cynb cenEnNotify <SERVICE UUID> <CHAR UUID>

Example:

COMMAND: cynb cenEnNotify 0x180D 0x2A39<cr>

RESPONSE: <cr_lf>

[12] [34]<cr_lf>

[56] [78]<cr_lf>

……

<cr_lf>

DISABLE NOTIFICATION

Function: BLE Central disable notification of specific characteristics.

Command Format: cynb cenDisNotify <SERVICE UUID> <CHAR UUID>

Example:

COMMAND: cynb cenDisNotify 0x180D 0x2A39<cr>

RESPONSE: <cr_lf>

OK;<cr_lf>

<cr_lf>

3.15 BLE Scan Start

PERFORM BLE SCAN

Function: Start Scanning

Command Format: cynb scanStart <INTERVAL> <WINDOW> <TIMEOUT>

Example:

COMMAND: cynb scanStart <cr>

RESPONSE: Start Scan<cr_lf>

GOLiFE CARE,ADV,[ED F1 9F 9B C7 31],-95,0;<cr_lf>

<cr_lf>

Note:

<Input Parameters> Interval value should be larger or equal to window value. The unit of INTERVAL and WINDOW is minisecond; The unit of TIMEOUT is second, which means that BLE Scanning will stop after <TIMEOUT> seconds, if <TIMEOUT> set to 0, disable timeout.

Default interval value is 500 ms, default window value is 400 ms, default time-out value is 5 seconds.
INTERVAL range: Maximum: 10240 Minimum:3
WINDOW range: Maximum: 10240 Minimum:3
TIMEOUT range: Maximum: 16383 Minimum:1
<Output Parameters>
<DEVICE_NAME>, ADV,<BLE ADDR>,<RSSI>,<ADV TYPE>,
<ADV TYPE> available list as below:
0:ADV_CONNECTABLE_UNDIRECTED,
1:ADV_CONNECTABLE_DIRECTED,
2:ADV_SCANNABLE_UNDIRECTED,
3:ADV_NON_CONNECTABLE_UNDIRECTED

3.16 BLE Scan Stop

STOP BLE SCAN

Function: Stop scanning BLE device.

Command Format: cynb scanStop

Example:

COMMAND: cynb scanStop <cr>
RESPONSE: <cr_lf>
Stop Scanning;<cr_lf>
<cr_lf>

3.17 BLE Connect

PERFORM BLE CONNECT

Function: Connect to specific BLE device by device address.

Command Format: cynb connect <ADDR>

Example:

COMMAND: cynb connect 0XE4FEAD218F5B<cr>

RESPONSE: serviceDiscoveryCallback<cr_lf>
S UUID-1800 attrs[1 7]<cr_lf>
C UUID-2a00 valueAttr[3] props[0] <cr_lf>
C UUID-2a01 valueAttr[5] props[0] <cr_lf>
C UUID-2a04 valueAttr[7] props[0] <cr_lf>
<cr_lf>

Note: Connect command will trigger service/characteristic discovery function after connection success.

3.18 BLE Disconnect

BLE DISCONNECTION

Function: Disconnect from current BLE connection.

Command Format: cynb disconn

Example:

COMMAND: cynb disconn<cr>
RESPONSE: <cr_lf>
OK<cr_lf>
<cr_lf>
Disconnected<cr_lf>
<cr_lf>

3.19 BLE Data Interrupt

BLE ENABLE DATA INTERRUPT

Function: Enable client write detection, action included showing write data, give interrupt to GPIO PIN and identify write command type.

Command Format: cynb enInt

Example:

COMMAND: cynb enInt<cr>
RESPONSE: <cr_lf>
OK<cr_lf>
<cr_lf>
...
w2,180F,2A19,1,22; <cr_lf>

RESULT: This example print string list in below sequence, (wX: Write command type), (Service UUID), (Characteristic UUID), (Data length), (Data in Hex)

Note:

Write command can be listed as below:
w0: Invalid operation
w1: Write
w2: Write without response
w3: Signed write
w4: Prepare write
w5: Cancel all prepared write
w6: Execute all prepared write

BLE DISABLE DATA INTERRUPT

Function: Disable client write detection

Command Format: cynb disInt

Example:

COMMAND: cynb disInt<cr>
RESPONSE: <cr_lf>
OK<cr_lf>
<cr_lf>

4. BLE Example for creating a profile by command

4.1 Tools Preparation

1. **Terminal tool:** Because the command line interface uses UART for configuration and communication, you need to install terminal emulator tool such as TERATERM or PUTTY.

2. **USB to UART convertor:** Connect the convertor to 1.8 V or 3.3 V UART output pin.

3. **BLE APP:** To display the BLE GATT profile created by CLI, you need to install BLE APP (in mobile phone or other BLE central system, at least support BLE 4.0) to verify the configuration and input data. Here we use Nordic's Wireless APP name "nRF Master Control Panel", which can be found on GOOGLE PLAY store for Android system, and APPLE STORE for iOS system.

4.2 Procedure for connecting UART

Use the terminal to choose the correct COM port (the information show on device manager) and set the data rate, also set the local echo on as figure 1 and figure 2. Press the reset button in module and reset module.

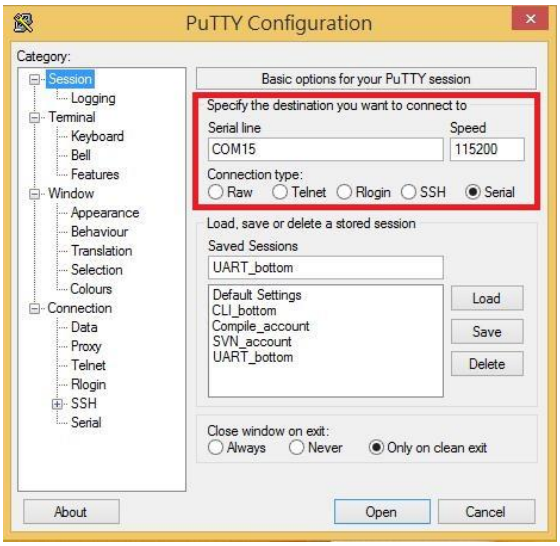

Figure 1. Terminal emulator configuration-1

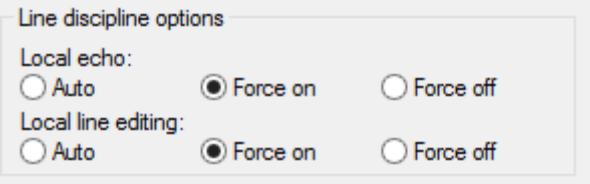

Figure 2. Terminal emulator configuration-2

Note: For other com port terminal software tool, enter string may not include "\r", therefore user need add it in the end of line.

4.3 Procedure for creating an Profile

BLE sensor device (in this case means Delta platform with glucose sensor) have 6 possible states, when module power on, transit to Initializing state.
After initialization, device transit to Configure state, in this state, Collector can configure device name and BLE address and build desired GATT services, etc.

RF is powered off to conserve energy for a specific time internal (defined in SIG profile) before transit to Advertising state where RF is powered on to send out an advertising packet. Once the transmission of advertising packet is completed, it transit back to Standby. The transition between Standby and Advertising will repeat until the sensor device is connected and transit to Connecting state when the advertising packet is received by central (collector) who intent to connect to this BLE sensor device.

In connecting state, all services such as Glucose service and Device Information Service can be accessed by central. The transition goes back to Standby when the connection is terminated by central or sensor device.

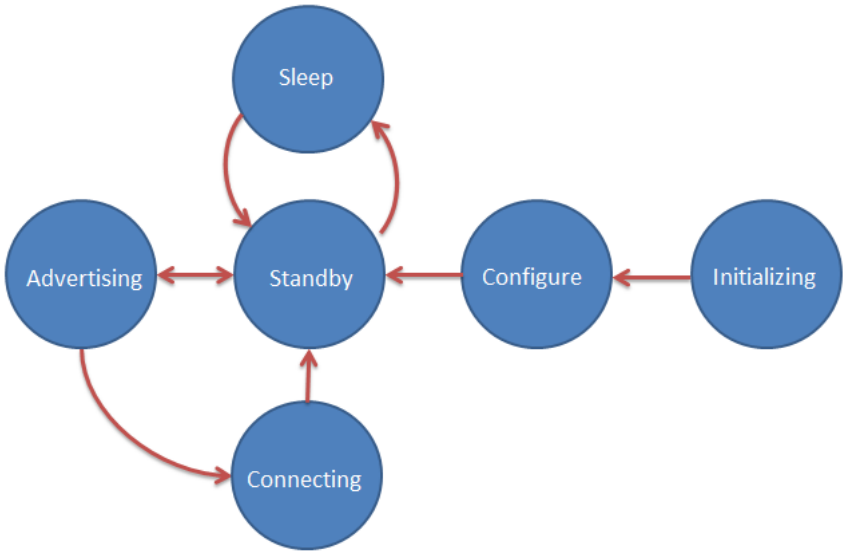

Figure.2 BLE state change diagram

Step1. Sensor power on, transit to **Initializing state**. Typically with software reset first, then check the module information and firmware information.

cynb reset

cynb info

Step2. Device transit to **Configure state**, configure by collector, with device name and transmit power, user can also build GATT services and characteristics, here 0x1808 correspond to Glucose Service, and Number string after correspond to their characteristics, and data format refer to Reference.1 “Glucose Profile”. On the other hand, user may add own vendor-specific services/characteristics UUID in this step.

cynb name <name length>s <device name string>

cynb bleAddr <BLE address 12 Hex>

cynb txPow <Integer Value>

cynb gattService 0x1808

cynb gattChar 0x2A34 <Property 2 Hex> <Hex Value>

cynb gattChar 0x2A51<Property 2 Hex> < Hex Value >

cynb gattChar 0x2A52 <Property 2 Hex> < Hex Value >

……

cynb regService

Step3. Glucose Sensor will typically remain powered off between uses, after sleep command execute, device transit to **Sleep state**, and will only advertise and allow a Collector to connect when it is turned on by the user and has data to send, key in any input will make device transit to **Standby state**.

cynb sleep

<any key to turn on>

Step4. Enter GAP connectable mode, device transit to **Advertising state**, sensor start advertising with specific interval and timeout, device waits until Collector initialize connection request.

cynb advStart 64 180

Step5. When device transit to **Connecting State**, meanwhile connection is established, the Glucose Sensor sends one or more notifications and indications to the Collector, also, if the property with write, Collector can initiatively update stored value in Sensor. User may also use BLE data interrupt function, inform Host for BLE data update from connected BLE device.

cynb update 0x1808 0x2A18 <TYPE> <Value>

cynb readData 0x1808 0x2A18

cynb enInt

Step6. When the data transfer is complete the Glucose Sensor typically terminates the connection, device back transit to **Standby state**.

cynb disconn

5. BLE Command Set Summary Table

Command Format	Command Response	Description
cynb initBleCen	OK	Init BLE Central stack
cynb gpio <GPIO-NO> <SET/CLEAR>	OK	Set or reset (high/low) a GPIO pin
cynb sleep <GPIO-NO>	OK	Go to system off mode, wake up by GPIO pin
cynb reset	OK	Perform soft reset for BLE module
cynb info	OK;<FW version>;<module name>;	Get BLE module Information
cynb name <LENGTH> <NAME>	OK	Set device name for BLE module
cynb name	OK;<device name>;	Get device name for BLE module
cynb txPow <TX POWER>	OK	Set Tx power to BLE module
cynb bleAddr <BLE ADDR>	OK	Set BLE address
cynb bleAddr	OK;<BLE ADDR>;	Get BLE address
cynb advStart <INTERVAL> <TIMEOUT>	OK	Start broadcast advertising packet with specific interval and timeout
cynb advStop	OK	Stop broadcast advertising packet
cynb gattService <SERVICE UUID>	OK	Add GATT service configuration
cynb gattChar <CHAR UUID> <PROPERTY> <VALUE>	OK	Add GATT characteristic in current add service
cynb regService	OK	Register new GATT service
cynb update <SERVICE UUID> <CHAR UUID> <TYPE> <VALUE>	OK;<Tx data>	Reset characteristic value to specific characteristics
cynb readData <SERVICE UUID> <CHAR UUID>	OK;<Rx data>	Read characteristic value of BLE module
cynb enInt	OK	Enable data interrupt
cynb disInt	OK	Disable data interrupt
cynb scanStart <INTERVAL> <WINDOW>	Start Scan; <DEVICE_NAME>,	Start to scan BLE device

<TIMEOUT>	ADV,<BLE ADDR>,<RSSI>,<ADV TYPE>	
cynb scanStop	Stop Scanning	Stop to scan BLE device
cynb connect <ADDR>	<SERVICE UUID>,<CHAR UUID>	Connect to specific BLE device by Device Address
cynb disconn	OK;Disconnected	Disconnect from current connection
cynb cenReadData <SERVICE UUID> <CHAR UUID>	OK;<Rx data>	BLE Central read value of specific characteristics
cynb cenWriteData <SERVICE UUID> <CHAR UUID> <TYPE> <VALUE>	OK;<Tx data>	BLE Central write value to specific characteristics
cynb cenEnNotify <SERVICE UUID> <CHAR UUID>	OK;<notification data>	BLE Central enable notification of specific characteristics
cynb cenDisNotify <SERVICE UUID> <CHAR UUID>	OK	BLE Central disable notification of specific characteristics

6. Simplified Command Set

6.1 Introduction

For code developer, repeated command test and usage is required, the time consuming is very large when keying the long command with 10 or higher alphabets. So the simplified version commands can solve this problem, it's defined and can be modified in source code "core_cli.cpp".

6.2 BLE Command Correspondence

GENERAL	
cynb initBleCen	BLE CIN
cynb gpio	BLE GIO
cynb sleep	BLE SLP
cynb reset	BLE RST
cynb info	BLE INF

cynb txPow	BLE POW
cynb name	BLE NAM
GATT	
cynb regService	BLE GRS
cynb gattChar	BLE GAC
cynb gattService	BLE GAS
cynb advStart	BLE ADS
cynb advStop	BLE ADP
cynb scanStart	BLE SCS
cynb scanStop	BLE SCP
cynb connect	BLE CON
cynb disconn	BLE DCN
cynb bleAddr	BLE ADR
cynb update	BLE WRT
cynb readData	BLE RED
cynb enInt	BLE EDI
cynb disInt	BLE DDI
cynb cenReadData	BLE CRD
cynb cenWriteData	BLE CWD
cynb cenEnNotify	BLE CEN
cynb cenDisNotify	BLE CDN