Index:

1. Introduction .................................................................................................................. 6
   Supported Platform ......................................................................................................... 7
   Reference ......................................................................................................................... 7
2. Commands Status Responses .......................................................................................... 8
   2.1 OK .......................................................................................................................... 8
   2.2 ERROR ...................................................................................................................... 8
3. BLE Commands Description .......................................................................................... 10
   3.1 GPIO ......................................................................................................................... 10
   3.2 System Off ................................................................................................................. 10
   3.3 Reset ......................................................................................................................... 11
   3.4 System Information ................................................................................................... 12
   3.5 Device Name ............................................................................................................ 12
   3.6 Init BLE Central Mode ............................................................................................. 13
   3.7 BLE Start Advertising ............................................................................................. 14
   3.8 BLE Stop Advertising .............................................................................................. 14
   3.9 TX Power .................................................................................................................. 15
   3.10 BLE Address .......................................................................................................... 15
   3.11 BLE GATT Service ............................................................................................... 16
   3.12 GATT Write Characteristic Value ......................................................................... 18
   3.13 GATT Read Characteristic Value .......................................................................... 20
   3.14 GATT Notification ................................................................................................. 20
   3.15 BLE Scan Start ....................................................................................................... 21
   3.16 BLE Scan Stop ....................................................................................................... 22
   3.17 BLE Connect .......................................................................................................... 22
   3.18 BLE Disconnect ..................................................................................................... 23
   3.19 BLE Data Interrupt ................................................................................................. 24
4. BLE Example for creating a profile by command .......................................................... 26
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Tools Preparation</td>
<td>26</td>
</tr>
<tr>
<td>4.2</td>
<td>Procedure for connecting UART</td>
<td>26</td>
</tr>
<tr>
<td>4.3</td>
<td>Procedure for creating an Profile</td>
<td>27</td>
</tr>
<tr>
<td>5.</td>
<td>BLE Command Set Summary Table</td>
<td>30</td>
</tr>
<tr>
<td>6.</td>
<td>Simplified Command Set</td>
<td>31</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>31</td>
</tr>
<tr>
<td>6.2</td>
<td>BLE Command Correspondence</td>
<td>31</td>
</tr>
</tbody>
</table>
### Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason of change</th>
<th>Maker</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>Jan 11, 2015</td>
<td>Initial release</td>
<td>Gill Wei</td>
</tr>
<tr>
<td>1.1</td>
<td>Mar 2, 2015</td>
<td>Add Wi-Fi Command Set</td>
<td>Marcus Chiou</td>
</tr>
<tr>
<td>1.2</td>
<td>Mar 10, 2015</td>
<td>Add BLE profile overview to introduction and BLE example for creating the sample by command</td>
<td>Gill Wei</td>
</tr>
<tr>
<td>1.3</td>
<td>Mar 27, 2015</td>
<td>Modify WIFI Command description</td>
<td>Marcus Chiou</td>
</tr>
<tr>
<td>1.4</td>
<td>Mar 30, 2015</td>
<td>Able to add BLE GATT 128bit vendor specific services/characteristics, modify advertise and add service command format.</td>
<td>Gill Wei</td>
</tr>
<tr>
<td>1.5</td>
<td>Jun 11, 2015</td>
<td>Modify 5.2 Procedure for creating an Profile about command description</td>
<td>Gill Wei</td>
</tr>
<tr>
<td>1.6</td>
<td>Sep 4, 2015</td>
<td>Merge mbed (mbed,BLE_API, RF51822) library, modify BLE name command, Wifi TCP/UDP blocking description</td>
<td>Gill Wei</td>
</tr>
<tr>
<td>1.7</td>
<td>Dec 9, 2015</td>
<td>Update GPIO PIN define to EVB version 3 Add BLE central mode API, WIFI device API, simplified command set.</td>
<td>Gill Wei</td>
</tr>
<tr>
<td>1.8</td>
<td>Jan 25, 2016</td>
<td>Add BLE data interrupt command; Chap 6.5 Wi-Fi data receive interrupt</td>
<td>Gill Wei</td>
</tr>
<tr>
<td>1.9</td>
<td>Jan 26, 2016</td>
<td>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</td>
<td>Gill Wei</td>
</tr>
<tr>
<td>1.10</td>
<td>Feb 18, 2016</td>
<td>Modify BLE name, Wifi TCP send, UDP send format</td>
<td>Gill Wei</td>
</tr>
<tr>
<td>1.11</td>
<td>Feb 25, 2016</td>
<td>Modify TCP and UDP max data length</td>
<td>Gill Wei</td>
</tr>
<tr>
<td>1.12</td>
<td>April 25, 2016</td>
<td>Modify BLE Update command(add raw data type), Fix update</td>
<td>Gill Wei</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Changes</td>
<td>Author</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>1.13</td>
<td>April 25, 2016</td>
<td>Fix Characteristic property setting issue, and readData command value length issue</td>
<td>Gill Wei</td>
</tr>
<tr>
<td>1.14</td>
<td>Sep 8, 2016</td>
<td>Add BLE Central mode commands</td>
<td>Silvia Chen</td>
</tr>
<tr>
<td>1.15</td>
<td>Sep 21, 2016</td>
<td>Fix BLE connect command issue; Modify the document for whole BLE module</td>
<td>Silvia Chen</td>
</tr>
<tr>
<td>1.16</td>
<td>Oct 20, 2016</td>
<td>Update Section 3.1 and Section 3.2</td>
<td>Tsungta Wu</td>
</tr>
</tbody>
</table>
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.
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

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

<table>
<thead>
<tr>
<th>CONTROL GPIO PIN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Function:</strong> Set or Clear a specific GPIO pin on BLE module</td>
</tr>
<tr>
<td><strong>Command Format:</strong> <code>cynb gpio &lt;LED-NO&gt; &lt;SET/CLEAR&gt;</code></td>
</tr>
</tbody>
</table>
| **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

<table>
<thead>
<tr>
<th>SLEEP MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Function:</strong> Go to system off mode, and then choose a specific pin; detect pin state change for wake up.</td>
</tr>
<tr>
<td><strong>Command Format:</strong> <code>cynb sleep &lt;BUTTON-NO&gt;</code></td>
</tr>
</tbody>
</table>
| **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, BUTTON2, BUTTON3 and BUTTON4 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`
- **RESPONSE:**
  
  OK; DELTA_CLI_V1.7;DFCM-NNN40-DT0R; `<cr>`

### 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`
- **RESPONSE:**
  
  OK; DELTA CLI `<cr>`

**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.”
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:
### WRITE CHARACTERISTIC 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 CHARACTERISTIC 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_if>
<cr_lf>`

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

**READ CHARACTERISTIC 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
S UUID-1800 attrs[1 7]
C UUID-2a00 valueAttr[3] props[0]
C UUID-2a01 valueAttr[5] props[0]
C UUID-2a04 valueAttr[7] props[0]

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<br>
RESPONSE: <cr_lf><cr_lf>OK<br><cr_lf>Disconnected
### 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.

![PuTTY Configuration](image)

**Figure 1. Terminal emulator configuration-1**
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

Step 1. Sensor power on, transit to **Initializing state**. Typically with software reset first, then check the module information and firmware information.

- `cynb reset`
- `cynb info`

Step 2. 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>`
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

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cynb scanStop</td>
<td>Stop Scanning</td>
</tr>
<tr>
<td>cynb connect &lt;ADDR&gt;</td>
<td>Connect to specific BLE device by Device Address</td>
</tr>
<tr>
<td>cynb disconn</td>
<td>Disconnect from current connection</td>
</tr>
<tr>
<td>cynb cenReadData &lt;SERVICE UUID&gt; &lt;CHAR UUID&gt;</td>
<td>BLE Central read value of specific characteristics</td>
</tr>
<tr>
<td>cynb cenWriteData &lt;SERVICE UUID&gt; &lt;CHAR UUID&gt; &lt;TYPE&gt; &lt;VALUE&gt;</td>
<td>BLE Central write value to specific characteristics</td>
</tr>
<tr>
<td>cynb cenEnNotify &lt;SERVICE UUID&gt; &lt;CHAR UUID&gt;</td>
<td>BLE Central enable notification of specific characteristics</td>
</tr>
<tr>
<td>cynb cenDisNotify &lt;SERVICE UUID&gt; &lt;CHAR UUID&gt;</td>
<td>BLE Central disable notification of specific characteristics</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>GENERAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>cynb initBleCen</td>
</tr>
<tr>
<td>cynb gpio</td>
</tr>
<tr>
<td>cynb sleep</td>
</tr>
<tr>
<td>cynb reset</td>
</tr>
<tr>
<td>cynb info</td>
</tr>
<tr>
<td>Command</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>cynb txPow</td>
</tr>
<tr>
<td>cynb name</td>
</tr>
<tr>
<td><strong>GATT</strong></td>
</tr>
<tr>
<td>cynb regService</td>
</tr>
<tr>
<td>cynb gattChar</td>
</tr>
<tr>
<td>cynb gattService</td>
</tr>
<tr>
<td>cynb advStart</td>
</tr>
<tr>
<td>cynb advStop</td>
</tr>
<tr>
<td>cynb scanStart</td>
</tr>
<tr>
<td>cynb scanStop</td>
</tr>
<tr>
<td>cynb connect</td>
</tr>
<tr>
<td>cynb disconn</td>
</tr>
<tr>
<td>cynb bleAddr</td>
</tr>
<tr>
<td>cynb update</td>
</tr>
<tr>
<td>cynb readData</td>
</tr>
<tr>
<td>cynb enInt</td>
</tr>
<tr>
<td>cynb disInt</td>
</tr>
<tr>
<td>cynb cenReadData</td>
</tr>
<tr>
<td>cynb cenWriteData</td>
</tr>
<tr>
<td>cynb cenEnNotify</td>
</tr>
<tr>
<td>cynb cenDisNotify</td>
</tr>
</tbody>
</table>