GR551x Device Synchronization Profile Example Application

Version: 1.3

Release Date: 2022-02-20

Shenzhen Goodix Technology Co., Ltd.

Trademarks and Permissions

GOODiX and other Goodix trademarks are trademarks of Shenzhen Goodix Technology Co., Ltd. All other trademarks and trade names mentioned in this document are the property of their respective holders.

Disclaimer

Information contained in this document is intended for your convenience only and is subject to change without prior notice. It is your responsibility to ensure its application complies with technical specifications.

Shenzhen Goodix Technology Co., Ltd. (hereafter referred to as “Goodix”) makes no representation or guarantee for this information, express or implied, oral or written, statutory or otherwise, including but not limited to representation or guarantee for its application, quality, performance, merchantability or fitness for a particular purpose. Goodix shall assume no responsibility for this information and relevant consequences arising out of the use of such information.

Without written consent of Goodix, it is prohibited to use Goodix products as critical components in any life support system. Under the protection of Goodix intellectual property rights, no license may be transferred implicitly or by any other means.

Shenzhen Goodix Technology Co., Ltd.

Headquarters: 2F. & 13F., Tower B, Tengfei Industrial Building, Futian Free Trade Zone, Shenzhen, China

TEL: +86-755-33338828 FAX: +86-755-33338099

Website: www.goodix.com

Preface

Purpose

This document introduces how to use and verify the Device Synchronization Profile example in the GR551x Software Development Kit (SDK), to help users quickly get started with secondary development.

Audience

This document is intended for:

- GR551x user
- GR551x developer
- GR551x tester
- Hobbyist developer
- Technical writer

Release Notes

This document is the fourth release of *GR551x Device Synchronization Profile Example Application*, corresponding to GR551x System-on-Chip (SoC) series.

Revision History

Version	Date	Description
1.0	2021-02-05	Initial release
1.1	2021-06-22	Updated the value range of Device Sync Event Period and relevant GRToolbox interfaces.
1.2	2021-08-09	Changed the section "Supported Development Platform" into "Preparation".
1.3	2022-02-20	Modified the file name of the example firmware based on SDK changes.

Contents

1 Introduction

Goodix provides a custom Device Synchronization Profile (DSP) example that enables synchronization of timing sequences between devices through parameter modification on a mobile App (GRToolbox). This means that a group of devices can operate in synchronous mode under one timing source.

This document introduces how to use and verify the DSP example in a GR551x Software Development Kit (SDK).

Before getting started, you can refer to the following documents.

Table 1-1 Reference documents

Name	Description
GR551x Developer Guide	Introduces GR551x SDK and how to develop and debug applications based on the SDK.
Bluetooth Core Spec	Offers official Bluetooth standards and core specification from Bluetooth SIG.
J-Link/J-Trace User Guide	Provides J-Link operational instructions. Available at www.segger.com/downloads/jlink/UM08001_JLink.pdf.
Keil User Guide	Offers detailed Keil operational instructions. Available at www.keil.com/support/man/docs/uv4/.
GR551x Bluetooth Low Energy Stack User Guide	Introduces the Bluetooth LE Protocol Stack supported by GR551x System-on-Chips (SoCs).

2 Profile Overview

This chapter introduces the device roles and Device Synchronization Service (DSS) that are defined by the DSP.

2.1 Device Roles

The DSP defines two roles: DS Host and DS Device.

- DS Host

  A DS Host is used to control the behavior of a DS Device by configuring its parameters over GRToolbox. The DS Host serves as the GAP Central, responsible for scanning for and connecting to a DS Device. Upon connection, the DS Host can receive timing synchronization information sent by the DS Device and control the behavior of the DS Device through commands. Such behavior includes setting a DS Device role, creating a timing source, enabling synchronization of timing sources, and deleting a timing source.

- DS Device

  A DS Device performs operations relevant to timing synchronization based on configurations set by the DS Host. During the process, in addition to being connected to the DS Host to receive control commands from the Host, the DS Device needs to connect to other DS Devices for timing synchronization.

  The interconnected DS Devices can be divided into two types based on different roles: Sync Source Role and Sync Device Role. A Sync Source, operating as both GAP Central and GAP Peripheral, is responsible for providing a timing source for its connected Sync Devices and synchronizing timing sequences between the Sync Device(s). A Sync Device serves as the GAP Peripheral only and waits for synchronization performed by the Sync Source.

The application scenario where GRToolbox is used as a DS Host is shown in Figure 2-1.

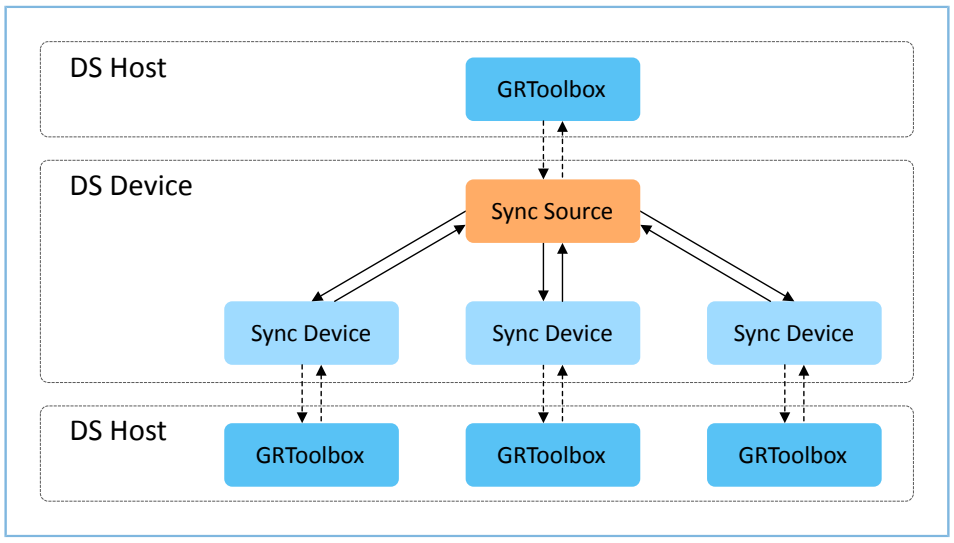

Figure 2-1 Application scenario

The following concepts can help you better understand how a DS Host interacts with a DS Device:

- Device Sync Event Count: the count number of timing synchronization events. After setting a device role and creating a timing source, the Sync Source generates a timing synchronization event every Device Sync Event Period. Device Sync Event Count starts from 0. In interaction scenarios where Auto Drift Calibration is enabled,

the Device Sync Event Count value can be used to decide on the time for calibration between the Sync Source and Sync Devices.

- Device Sync Event Period: an interval between synchronization events in units of 312.5 µs. After being synchronized by the Sync Source, the Sync Device generates a timing synchronization event at regular intervals in line with the Sync Source. The Device Sync Event Period shall be neither too short nor too long (for a proper value range, see "Chapter 6 Appendix"). A value smaller than the lower threshold results in connection failure between a Sync Source and a Sync Device due to the insufficient interval between two sync events. In scenarios with a Device Sync Event Period longer than the upper threshold, the difference between the event period for the Sync Device and that for the Sync Source is excessively large even after calibration.
- Auto Drift Calibration Period: a period for automatic drift calibration. Prior to synchronization, it is recommended to enable Auto Drift Calibration (see "Chapter 6 Appendix") and set the Auto Drift Calibration Period to a value that equals the number of synchronization events. This helps decrease the period difference caused by timing drifts. Within an Auto Drift Calibration Period, one round of drift calibration is performed on both the Sync Source and Sync Device.

  Since the Auto Drift Calibration Period value is identical with Device Sync Event Count, Device Sync Event Period should be taken into consideration when Auto Drift Calibration Period is set. Neither an excessively high value nor an excessively low value is allowed. When the Device Sync Event Period is relatively short, the Auto Drift Calibration Period should not be set to an excessively small value; otherwise the Sync Source cannot finish synchronization with the Sync Device within the Auto Drift Calibration Period. When the Device Sync Event Period is relatively long, the Auto Drift Calibration Period should not be set to an excessively large value; otherwise, the difference between the sync event period of the Sync Source and that of the Sync Device is excessively large before calibration, undermining timing synchronization performance.

When a DS Device receives a timing synchronization event, the voltage of the pin used to check whether a synchronization event has been received will be toggled twice to verify the timing synchronization results. During the process, you can see the synchronization events displayed as pulses on a logic analyzer. When a Sync Source completes creating a timing source, you can see a pulse every Device Sync Event Period.

After a Sync Source starts synchronization or drift calibration with a Sync Device, the Sync Device initiates advertising while the Sync Source connects to the device after finding its advertisement during scanning. Upon connection, the Sync Device starts to receive timing synchronization information delivered by the Sync Source. Based on the received information, the Sync Device decides on or adjusts the synchronization time point. Throughout the process, no pulse occurs because no timing synchronization event has been generated on the Sync Device.

Figure 2-2 shows a detailed synchronization process where Auto Drift Calibration is enabled for both the Sync Source and Sync Device with the Auto Drift Calibration Period set to 10.

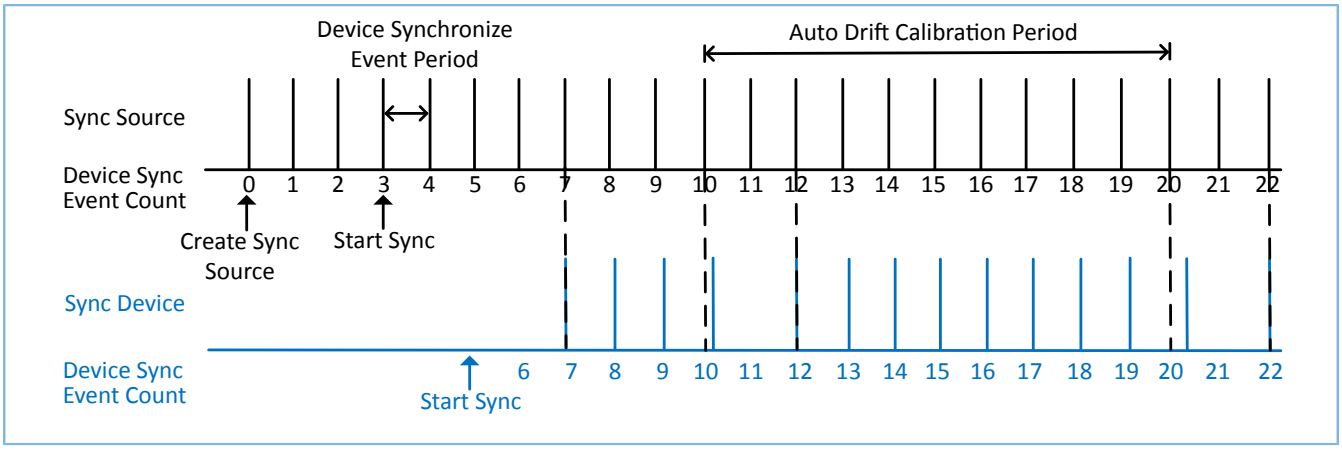

Figure 2-2 Timing synchronization process

2.2 Advertising Data

The DSP defines two types of advertisement: Goodix_DSS_CFG and Goodix_DSS_SYNC.

- Goodix_DSS_CFG: responsible for connection between DS Host and DS Device
- Goodix_DSS_SYNC: responsible for connection between DS Devices, facilitating timing synchronization

The DS Device decides on which advertisement to be enabled based on the actual scenario.

2.3 Device Synchronization Service

The Device Synchronization Service (DSS), customized in Goodix DSP, transmits data and commands as well as receives responses related to timing synchronization. The 128-bit vendor-specific UUID of DSS is A6ED0A01-D344-460A-8075-B9E8EC90D71B.

Table 2-1 lists characteristics of DSS.

Table 2-1 DSS characteristics

Characteristic	UUID	Type	Support	Security	Property
Device Sync Role	A6ED0A02-D344-460A-8075-B9E8EC90D71B	128 bits	Mandatory	None	Read
Device Sync Event Count	A6ED0A03-D344-460A-8075-B9E8EC90D71B	128 bits	Mandatory	None	Read, Notify
Device Sync Event Period	A6ED0A04-D344-460A-8075-B9E8EC90D71B	128 bits	Mandatory	None	Read
Device Sync Status	A6ED0A05-D344-460A-8075-B9E8EC90D71B	128 bits	Mandatory	None	Read
Device Sync Control Point	A6ED0A06-D344-460A-8075-B9E8EC90D71B	128 bits	Mandatory	None	Write, Indicate

2.3.1 Device Sync Role Characteristic

When the Device Sync Role characteristic is read, device role information of the relevant DS Device will be returned. The value of the Device Sync Role characteristic is defined by a DS Host during device role configuration (for details, see "Chapter 6 Appendix"). By default, the value of Device Sync Role is set to **Invalid Role**.

2.3.2 Device Sync Event Count Characteristic

When the Device Sync Event Count characteristic is read or its notification function is enabled, a Device Sync Event Count value that indicates the count number of timing synchronization events will be returned.

2.3.3 Device Sync Event Period Characteristic

When the Device Sync Event Period characteristic is read, a Device Sync Event Period value in units of 312.5 µs will be returned.

- When DS Device is Sync Source Role, this characteristic value is defined when a DS Host creates a timing source (for details, see "Chapter 6 Appendix").
- When DS Device is Sync Device Role, this characteristic value is synchronized to the Device Sync Event Period of Sync Source during the DS Host synchronization process.

2.3.4 Device Sync Status Characteristic

When the Device Sync Status characteristic is read, a status value, indicating the timing synchronization status, will be returned. Status values include Config Ready (0x00), In Adv (0x01), In Scan (0x02), and In Initiating (0x03), as illustrated in Figure 2-3.

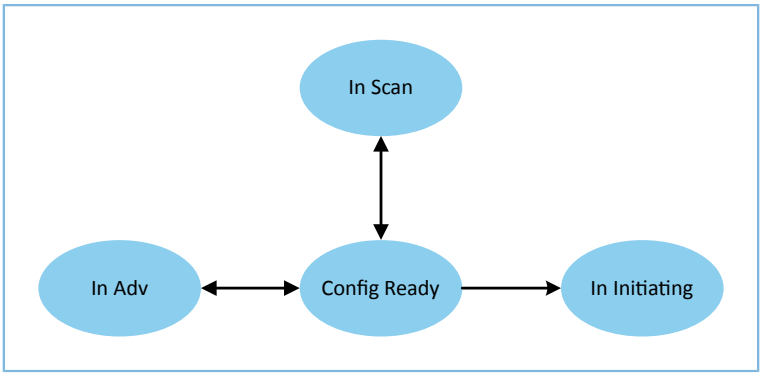

Figure 2-3 Timing synchronization state machines

2.3.5 Device Sync Control Point Characteristic

Through the Device Sync Control Point characteristic, a DS Host controls a DS Device to perform specific operations.

Format of the Device Sync Control Point characteristic: Opcode + Parameter. For detailed definitions, see "Chapter 6 Appendix".

3 Initial Operation

This chapter introduces how to run and verify a DSP example in a GR551x SDK.

Note:

SDK_Folder is the root directory of GR551x SDK.

3.1 Preparation

Perform the following tasks before running a DSP example.

- **Hardware preparation**

Table 3-1 Hardware preparation

Name	Description
Development board	GR5515 Starter Kit Board (SK Board). Three boards are required.
Connection cable	Micro USB 2.0 cable
Android phone	A mobile phone running on Android 5.0 (KitKat) and later

- **Software preparation**

Table 3-2 Software preparation

Name	Description
Windows	Windows 7/Windows 10
J-Link driver	A J-Link driver. Available at www.segger.com/downloads/jlink/.
Keil MDK5	An integrated development environment (IDE). MDK-ARM Version 5.20 or later is required. Available at www.keil.com/download/product/.
GRToolbox (Android)	A Bluetooth LE debugging tool. Available in SDK_Folder\tools\GRToolbox.
GRUart (Windows)	A serial port debugging tool. Available in SDK_Folder\tools\GRUart.
GProgrammer (Windows)	A programming tool. Available in SDK_Folder\tools\GProgrammer.

3.2 Firmware Programming

The source code of the DSP example is in SDK_Folder\projects\ble\ble_multi_role\ble_app_dss.

You can programme *ble_app_dss.bin* to three SK Boards though GProgrammer. All of the three boards serve as DS Devices, of which SK Board A operates in Sync Source Role while SK Board B and SK Board C operate in Sync Device Role. For details, see *GProgrammer User Manual*.

Prior to firmware programming, make sure the following has been implemented:

- The DSP project is demanding in the program running speed, so Mirror mode is recommended. Modify the value of **APP_CODE_RUN_ADDR** in *custom_config.h* on demand. In this document, **APP_CODE_RUN_ADDR** is modified to **0x00820000**.

- For a project involving modification to the source code of ble_app_dss, re-compile the project to generate a new *ble_app_dss.bin* file, and then programme the file to the three SK Boards. For details, refer to *GR551x Developer Guide*.

Note:

- The *ble_app_dss.bin* is in SDK_Folder\projects\ble\ble_multi_role\ble_app_dss\build.
- A device should be configured with an advertising address depending on its device role.

3.3 Test and Verification

Running a DSP example project requires at least two SK Boards. In this document, three SK Boards are used. Accordingly, three Android phones running GRToolbox are required to control the three DS Devices. When all required hardware and software resources are ready, test and verify the DSP example. Steps are described as follows:

1. Enable advertising on DS Devices, and use GRToolbox to scan the advertisement.

   Press the **OK** buttons on the three SK Boards. Start GRToolbox on the three phones. Tap **Application** > **DSS** to open the DSS control interface, as shown in Figure 3-1.

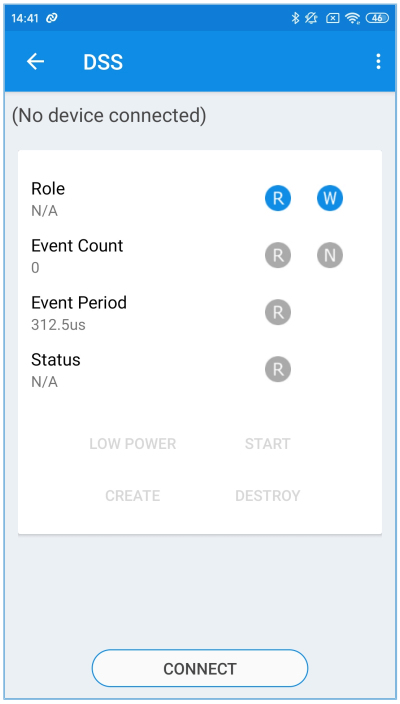

Figure 3-1 **DSS** control interface

Note:

GRToolbox screenshots in this document are used to help users better understand test and verification steps only. The user interface of GRToolbox in actual use prevails.

   On GRToolbox, tap **CONNECT** to start scanning, and each GRToolbox shows three devices with an advertising name of **Goodix_DSS_CFG** (the advertising name can be configured to other values in *user_app.c*) in Figure 3-2.

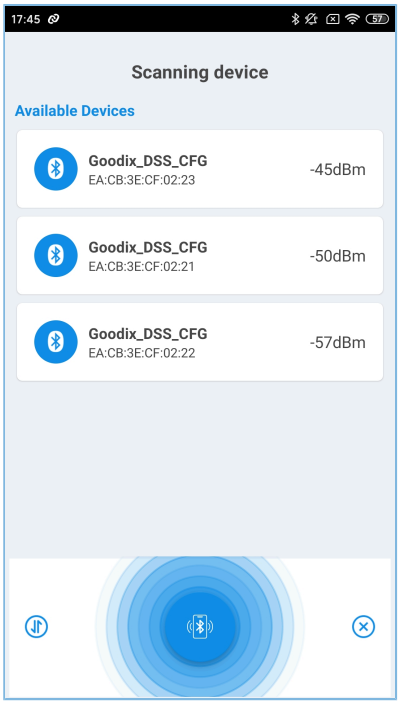

Figure 3-2 Discovering **Goodix_DSS_CFG**

2. Tap **Goodix_DSS_CFG** to connect the corresponding DS Device to GRToolbox.

   Connect one of the GRToolbox programs to one of the DS Devices advertising as **Goodix_DSS_CFG**. This means one GRToolbox software program connects to only one DS Device. Figure 3-3 presents the interface when connection has been established. On this interface, you can:

   - **Role**: Set and read a device role.
   - **Event Count**: Read Device Sync Event Count and enable its notification.
   - **Event Period**: Read Device Sync Event Period.
   - **Status**: Read Device Sync Status.
   - **LOW POWER**: Enter low-power mode.
   - **START**: Start synchronization.
   - **CREATE** and **DESTROY**: Create and delete a timing source.

   Note that only Sync Sources can perform **CREATE**.

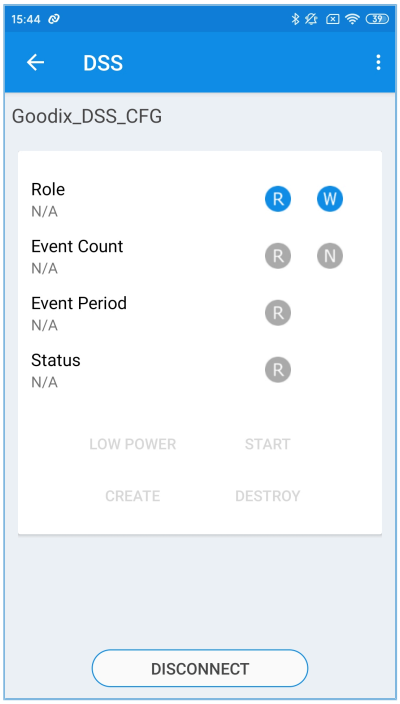

Figure 3-3 GRToolbox interface when connection has been established

3. Set a device role.

   As shown in Figure 3-4, on the GRToolbox connecting to SK Board A, tap **W** on the right side of **Role** > **Sync Source Role**. On the GRToolbox programs connecting to SK Board B and SK Board C, tap **W** on the right side of **Role**, and choose **Sync Device Role**. Tap **R** on the right side of **Role** to read the device role, as shown in Figure 3-5.

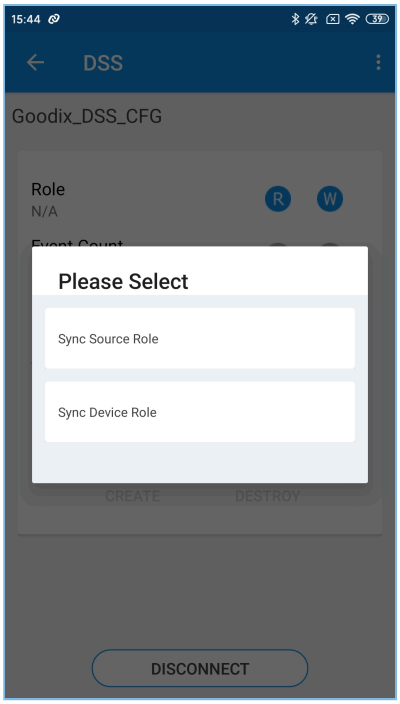

Figure 3-4 Setting a device role

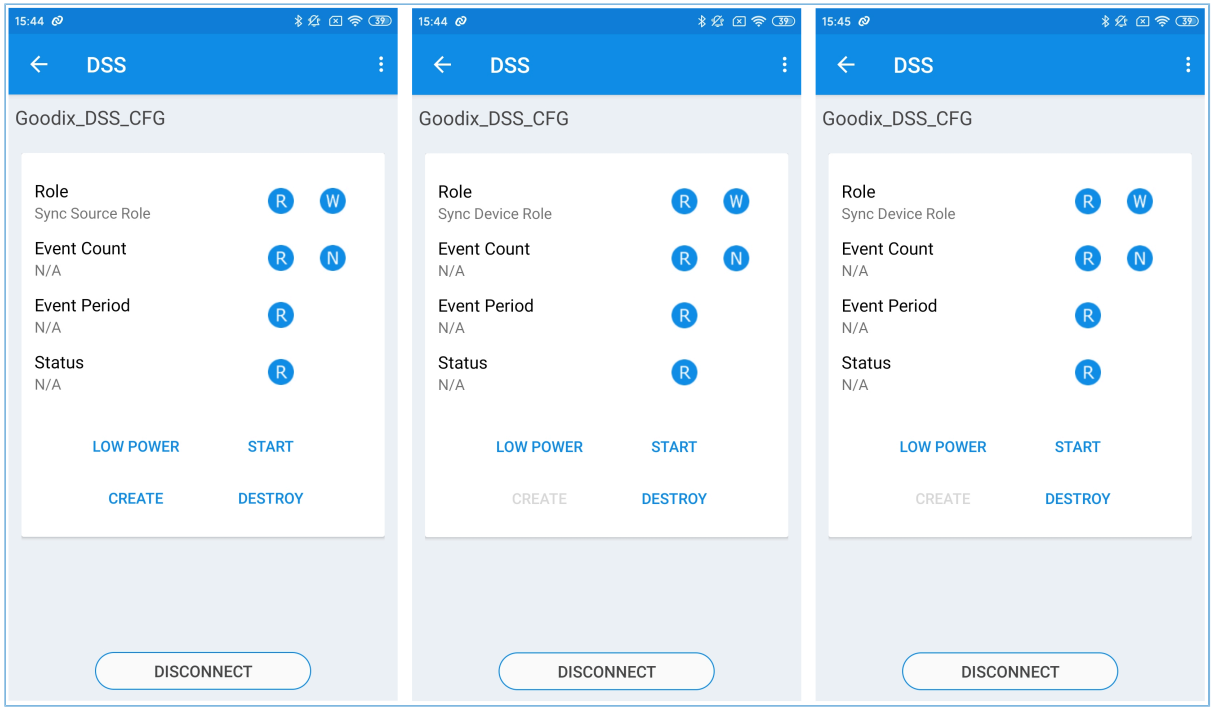

Figure 3-5 Reading device roles

4. Create a timing source.

   On the GRToolbox connecting to the Sync Source (SK Board A), tap **CREATE** to create a timing source. In this demo, the Device Sync Event Period is set to **3000**, which means the Device Sync Event Count increases by 1 every 0.9375s. Tap **R** on the right side of **Event Period**, you can see **3000 -> 937500.0us**, as illustrated in Figure

3-6. Tap **R** or **N** besides **Event Count**, and you can see the value under **Event Count** increases. Meanwhile, you can find that the **Synchronize count** increases on GRUart.

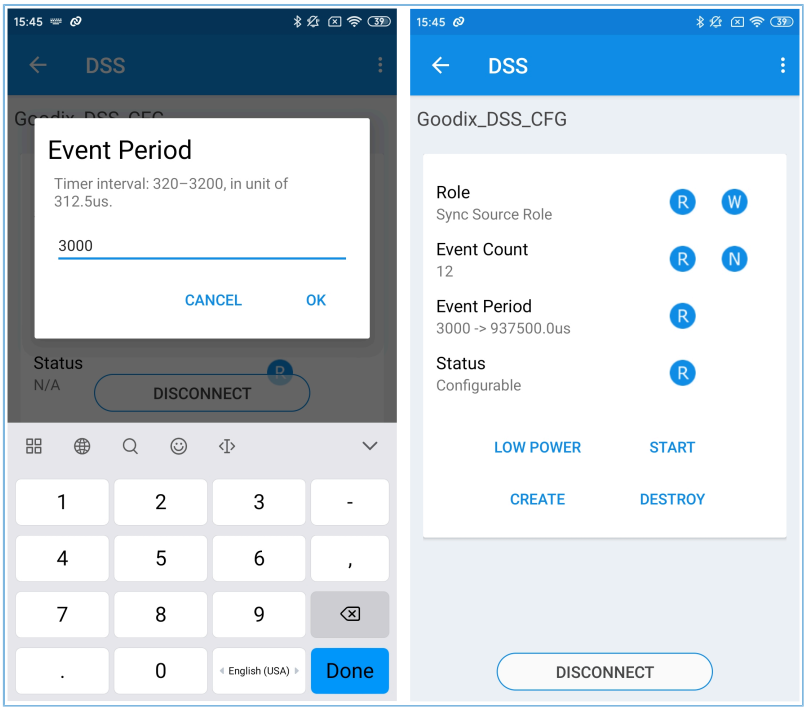

Figure 3-6 Creating a timing source

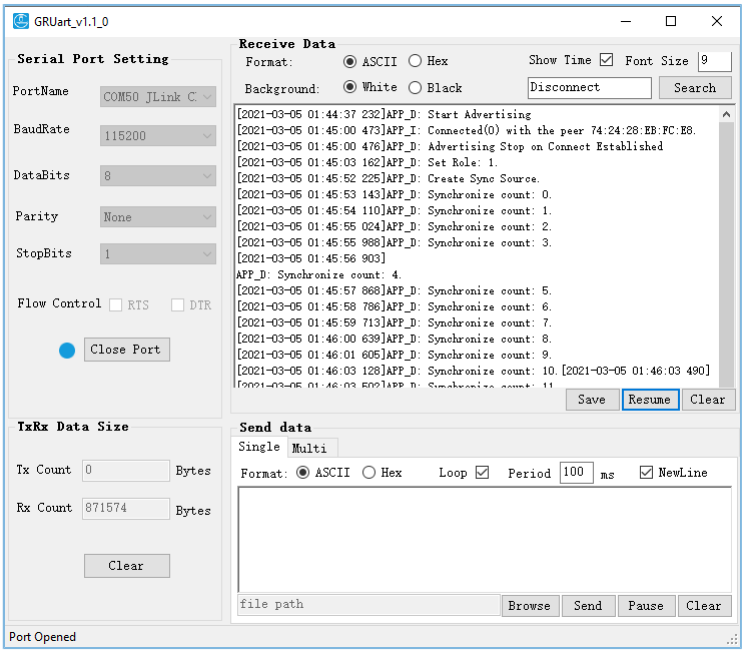

Figure 3-7 Increasing of Device Sync Event Count

5. Start synchronization.

(1). On the GRToolbox connecting to the Sync Source (SK Board A), tap **START** to start synchronization.

In this demo, both **Auto Drift Calibration** and **Auto Low Power Enter** are enabled. **Auto Drift Cali Period** is set to 10, which means a round of automatic drift calibration is performed every 10 sync events. **Auto Sync Device Num** is set to 2, which means up to two Sync Devices can be synchronized by the Sync Source.

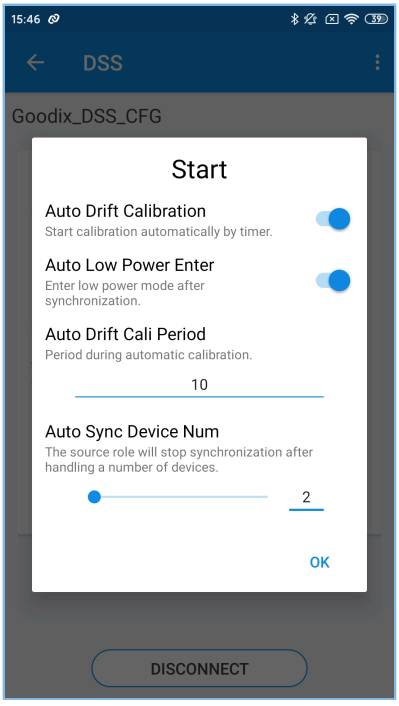

Figure 3-8 Setting synchronization parameters

(2). On the GRToolbox connecting to the two Sync Devices, tap **START**, and configure **Auto Drift Cali Period** to 10 and enable **Auto Drift Calibration** (it is recommended that Sync Devices share settings of **Auto Drift Cali Period** and **Auto Drift Calibration** with the Sync Source). For **Auto Low Power Enter**, there is no need to keep it the same as that for the Sync Source. You can choose to either enable or disable it. **Auto Sync Device Num** can be left blank.

(3). When configuration of mandatory parameters of all devices completes, the Sync Source starts synchronization with Sync Devices.

The Sync Source synchronizes data with one Sync Device after another. If **Auto Low Power Enter** is enabled for all devices, the two Sync Devices will enter low-power mode and be disconnected from GRToolbox immediately after synchronization completes. The Sync Source will enter low-power mode and be disconnected from both Sync Devices and GRToolbox after synchronization with both of the Sync Devices. Meanwhile, you can find that the **Synchronize count** increases on GRUart, as shown in Figure 3-9.

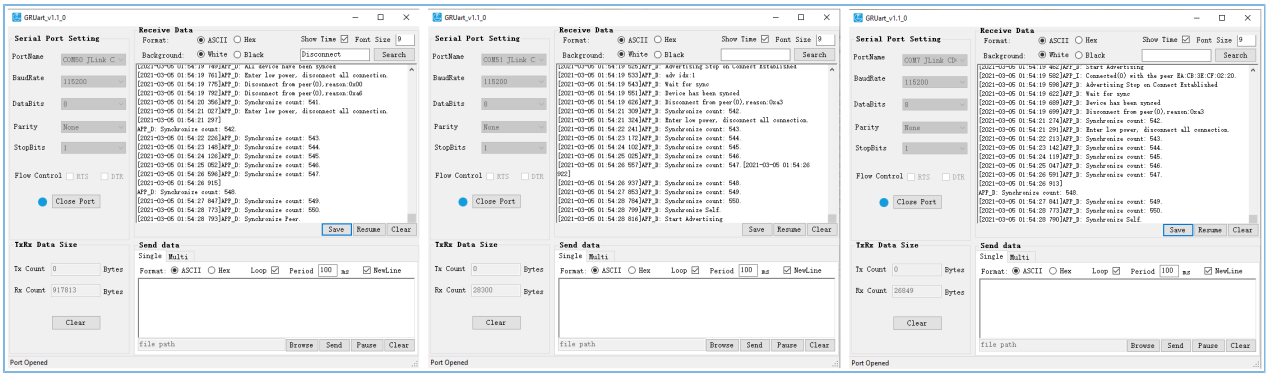

Figure 3-9 GRUart printing info

During the process, you can utilize a logic analyzer to observe the voltage level changes of GPIO_4, a pin for checking whether a synchronization event has been received by the Sync Devices (SK Board B and SK Board C). The captured data results are depicted in Figure 3-10.

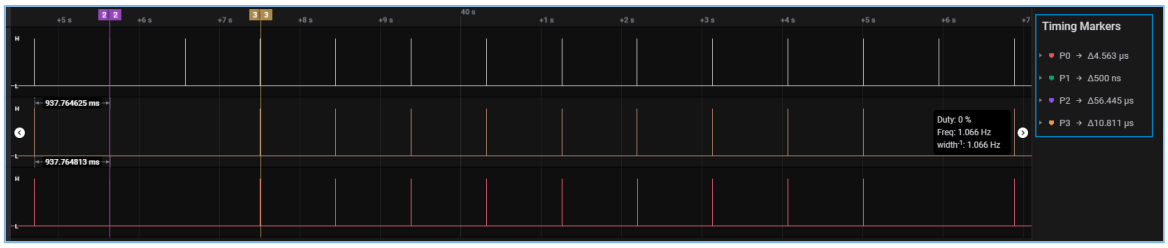

Figure 3-10 I/O level toggling information

T1, representing the time lag between the last timing synchronization event of the Sync Source and that of the Sync Device prior to automatic drift calibration, is the largest. T1 is affected by both Device Sync Event Period and Auto Drift Cali Period. High Device Sync Event Period and Auto Drift Cali Period values result in a larger T1. T2, representing the time lag between the first timing synchronization event of the Sync Source and that of the Sync Device after automatic drift calibration, is the smallest. Maximum value for T2 = Device Sync Event Period x 312.5

µs x Deviation of the SoC's crystal oscillator (ppm). In this demo, the deviation of the SoC's crystal oscillator is 500 ppm.

Utilize a logic analyzer to test T1 and T2. The test markers are displayed in Figure 3-10 and Figure 3-11. Test results are presented in Table 3-3.

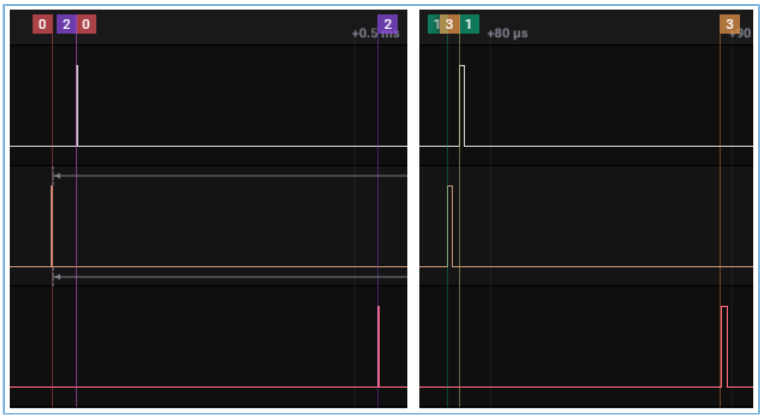

Figure 3-11 Test markers

Table 3-3 Time lags

Device Name	T1	T2
SK Board B	4.563 µs	500 ns
SK Board C	56.445 µs	10.811 µs

As shown in the table, when Device Sync Event Period = 3000 and Auto Drift Cali Period = 10, the T2 value representing the time lag between the first timing synchronization event of SK Board B and that of SK Board A after automatic drift calibration is 500 ns, and the T2 value for SK Board C is 10.811 µs. Both the obtained T2 values are smaller than the allowed maximum value of T2, 46.875 µs (3000 x 312.5 µs x 500 ppm).

6. Cancel synchronization.

   - After the Sync Source implements **START** in Step 5, if no Sync Device has been discovered after long-time scanning, tap **CANCEL** to cancel synchronization.
   - After the Sync Source starts synchronization with a Sync Device, if connection to the Sync Source fails after repeated retries, tap **CANCEL** to cancel synchronization.

7. Delete a timing source.

   On the GRToolbox connecting to the Sync Source (SK Board A), tap **DESTROY** to delete a timing source and stop synchronization.

8. Enter low-power mode.

   If **Auto Low Power Enter** is disabled for a DS Device after tapping **START** in Step 5, the DS Device will not enter low-power mode after synchronization. In this case, tap **LOW POWER** to enable the low-power mode.

4 Application Details

This chapter introduces the running procedures and major code of the DSP example.

4.1 Running Procedures

When the DSP example starts running, the system initializes peripherals, BLE Protocol Stack, and DSS successively.

Figure 4-1 shows a running procedure after a DS Host (GRToolbox in this document) connects to a DS Device after discovering its advertisement in scanning. To simplify demonstration, only one Sync Source and one Sync Device are used.

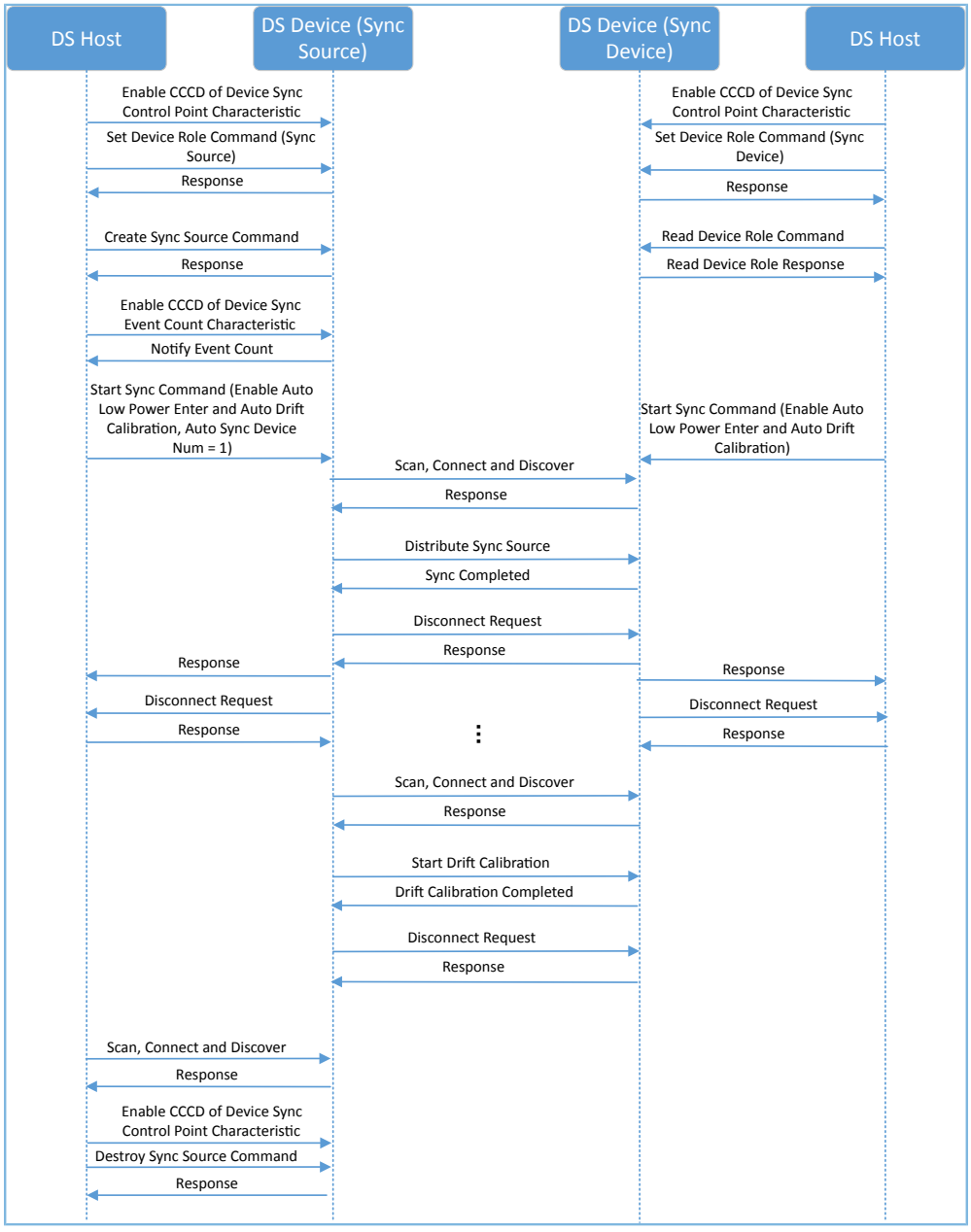

Figure 4-1 Running procedures

4.2 Major Code

The sections below depict major code relevant to interaction between DS Host and DS Devices (Sync Source and Sync Device).

4.2.1 Receiving Commands from DS Host

When a DS Device receives control command data from the DS Host, it parses the corresponding event, reports it to the application layer, and executes the corresponding command.

Path: `user_app\user_app.c` under the project directory

Name: dss_evt_handler();

```
static void dss_evt_handler(dss_evt_t *p_evt)
{
    ...
    if (DSS_EVT_SYNC_SELF_OR_PEER == p_evt->evt_type)
    {
          s_is_enter_lp_mode = p_evt->is_enter_lp_mode;

          if (DSS_ROLE_SYNC_DEVICE == s_role)
          {
              error_code |= ble_gap_adv_param_set(DSS_SYNC_ADV_IDX,
                  BLE_GAP_OWN_ADDR_STATIC, &s_gap_adv_param);
              error_code |= ble_gap_adv_data_set(DSS_SYNC_ADV_IDX,
                  BLE_GAP_ADV_DATA_TYPE_DATA, s_adv_data_sync, sizeof(s_adv_data_sync));
              error_code |= ble_gap_adv_start(DSS_SYNC_ADV_IDX, &s_gap_adv_time_param);
              s_is_in_adv = true;
          }
          ...
    }
    ...
}
```

4.2.2 Command for Starting Synchronization

DSS parses the command for starting synchronization and reports the **DSS_EVT_SYNC_SELF_OR_PEER** event to the application layer.

- If the relevant device is in Sync Source Role, start scanning through dss_evt_handler(). When any advertisement has been found during scanning, use app_adv_report_handler() to filter Sync Devices. When the Sync Source discovers its target device, stop scanning and establish connection to the device by using app_scan_stop_handler(). You can find all the relevant functions in *user_app.c*. Upon connection, the Sync Source can implement synchronization with the Sync Device by using dss_sync_src_distribute() in *dss.c*.
- If the relevant device is in Sync Device Role, start advertising through dss_evt_handler(). After being connected to the Sync Source, the Sync Device can be synchronized with the source by calling dss_sync_src_distribute() in app_adv_stop_handler() that is located in *user_app.c*.

The major code for starting synchronization of a Sync Source is presented below:

Path: `user_app\user_app.c` under the project directory

Name: app_connected_handler();


```
void app_connected_handler(uint8_t conn_idx, uint8_t status,const gap_conn_cmp_t
                           *p_conn_param)
{
    …
    dss_sync_src_distribute(conn_idx);
    dss_sync_op_result_send(s_cfg_conn_idx, s_evt_type, DSS_RSP_ID_SUCCESS);
    ble_gap_disconnect(conn_idx);
    …
}
```

4.2.3 Command for Cancelling Synchronization

DSS parses the command for cancelling synchronization and reports the **DSS_EVT_SYNC_CANCEL** event to the application layer.

- If the relevant device is in Sync Source Role, check whether the device is in scanning state. If yes, stop scanning and reset the parameters configured in Step 5: Start synchronization (see "Section 3.3 Test and Verification").
- If the relevant device is in Sync Device Role, check whether the device is in advertising state. If yes, stop advertising and reset the parameters configured in Step 5: Start synchronization (see "Section 3.3 Test and Verification").

The major code for cancelling synchronization of a Sync Source is presented below:

Path: `user_app\user_app.c` under the project directory

Name: dss_evt_handler();

```
static void dss_evt_handler(dss_evt_t *p_evt)
{
    if (DSS_EVT_SYNC_CANCEL == p_evt->evt_type)
    {
        …
        if (DSS_ROLE_SYNC_SOURCE == s_role)
        {
            if (s_is_in_scan)
            {
                if (ble_gap_scan_stop())
                {
                    rsp_id = DSS_RSP_ID_CANCEL_SYNC_FAIL;
                }
                else
                {
                    dss_reset_params(s_cfg_conn_idx);
                }
            }
        }
    }
    …
}
```

5 FAQ

This chapter describes possible problems, reasons, and solutions when using the DSP example.

5.1 Why Does Auto Drift Calibration Between Devices Fail?

- Description

  When tapping **START** to enable synchronization, **Auto Drift Calibration** is enabled for both the Sync Source and Sync Device. However, both Sync Source and Sync Device fail to perform automatic drift calibration when running to their Auto Drift Cali Period values. In addition, timeout information is returned.

- Analysis

  The **Auto Drift Cali Period** of the Sync Source is set to a value that is different from that of the Sync Device. In this case, automatic drift calibration occurs only when both the Sync Source and the Sync Device run to the higher **Auto Drift Period** value of the two devices.

- Solution

  Set the **Auto Drift Cali Period** of the Sync Source to a value which is identical with that of the Sync Device before implementing **START**.

6 Appendix

The Device Sync Control Point characteristic comes in a format of Opcode + Parameter. The table below provides the detailed definitions.

Table 6-1 Device Sync Control Point procedure definitions

Opcode	Parameter	Description
0x00	N/A	Invalid
0x01	<Device Sync Role(UINT8)> • 0x00: Invalid Role (default) • 0x01: Sync Source Role • 0x02: Sync Device Role	Initiates a procedure to set a device role for the DS Device. The parameter indicates a device role set by users. After successful setting, obtain the parameter value by reading the Device Sync Role characteristic.
0x02	<Device Sync Event Period (UINT16)> • Unit: 312.5 μs • Range: 0x0140 to 0x0CB0	Initiates a procedure to create a timing source. The parameter indicates the period for a synchronization event set by users. After successful setting, read the parameter value via the Device Sync Event Period characteristic. This procedure applies to Sync Source Role only.
0x03	<Auto Drift Calibration (UINT8 bit0)> • 0: Disable • 1: Enable <Auto Low Power Enter (UINT8 bit1)> • 0: Disable • 1: Enable <Auto Drift Calibration Period (UINT32)> • The parameter value equals the number of timing synchronization events. <Auto Sync Device Num (UINT8)> • Range: 1 to 30	Initiates a procedure to implement timing synchronization between DS Devices. The parameters that can be customized by users include: • Auto Drift Calibration • Auto Low Power Enter • Auto Drift Calibration Period • Auto Sync Device Num **Note:** The Auto Drift Calibration Period is valid only when Auto Drift Calibration is enabled. The Auto Drift Calibration Period value of the Sync Source should be identical with that of the Sync Device.
0x04	NULL	Initiates a procedure to cancel timing synchronization between DS Devices.
0x05	NULL	Initiates a procedure for the DS Device to enter low-power mode.
0x06	NULL	Initiates a procedure for the DS Device to delete a timing source and stop timing synchronization between DS Devices.
0xFF	<Request Op Code (UINT8) >& <Response value (UINT8)>	Labels responses to the Device Sync Control Point.