# GR533x LCP Developer Guide

**Version: 1.0**

**Release Date: 2023-10-15**

**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: Floor 12-13, Phase B, Tengfei Industrial Building, Futian Free Trade Zone, Shenzhen, China

TEL: +86-755-33338828 Zip Code: 518000

Website: www.goodix.com

# Preface

## Purpose

This document introduces the data packet format and application model of GR533x Light Communication Protocol (LCP, a Goodix 2.4 GHz proprietary protocol), to help users quickly get started with development of LCP-based communication applications.

## Audience

This document is intended for:

- Device user
- Software developer
- Test engineer

## Release Notes

This document is the initial release of *GR533x LCP Developer Guide*, corresponding to GR533x SDK.

## Revision History

| Version | Date | Description |
|---|---|---|
| 1.0 | 2023-10-15 | Initial release |

# Contents



 

# 1 Introduction

The GR533x System-on-Chip (SoC) integrates a 2.4 GHz RF transceiver, supporting Bluetooth Low Energy (Bluetooth LE) protocols and Light Communication Protocol (LCP, a Goodix 2.4 GHz proprietary protocol).

Main features of LCP:

- Operating frequency: even frequencies within 2360 MHz–2520 MHz
- Transfer rate: GFSK 1 Mbps/2 Mbps
- Support Cyclic redundancy check (CRC).
- Support Whitening.
- TX/RX mode:
  - One-shot software-triggered TX/RX mode
  - Periodic timer-triggered TX/RX mode

# 2 Data Packet Format

The LCP data packet format is shown below:

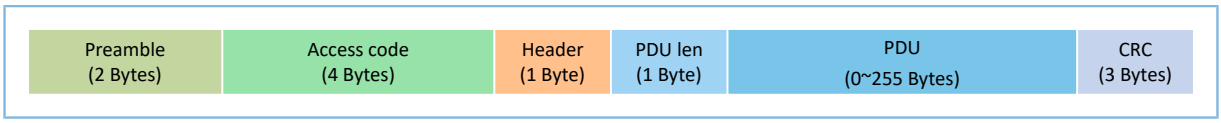


Figure 2-1 LCP data packet format

- Preamble: 2-byte preamble, with an alternate sequence of 0 and 1 automatically determined by the SDK package and the last bit of the alternate sequence being different from the first bit of the access code
- Access code: customized 4-byte access code for identifying devices. The receiver can only receive data with specified access code. GR533x only supports checking one access code at a time.
- Header: customized 1-byte header
- PDU len: 1-byte data packet length
- PDU: user data; range: 0–255 (unit: byte). The actual length is determined by PDU len.
- CRC: 3-byte cyclic redundancy check code; the CRC algorithm is the same as that specified in *Bluetooth Core Spec*. You can enable or disable CRC.

The aforementioned parameters, except for PDU, have fixed lengths and cannot be configured.

# 3 Application Model

This chapter depicts the communication modes, communication roles, and related APIs of LCP, as well as implementation procedures and reference examples of LCP applications.

## 3.1 Communication Modes

According to the RF behavior triggering source types, the LCP TX/RX modes can be divided into the following four categories:

- Software-triggered RX mode (LCP_TRX_MODE_SW_RX): The software calls the LCP RX API to receive data packets.
- Software-triggered TX mode (LCP_TRX_MODE_SW_TX): The software calls the LCP TX API to transmit data packets.
- Timer-triggered RX mode (LCP_TRX_MODE_TIMER_RX): The timer automatically triggers calling of the LCP RX API to receive data packets.
- Timer-triggered TX mode (LCP_TRX_MODE_TIMER_TX): The timer automatically triggers calling of the LCP TX API to transmit data packets.

---

**Note**:

Four timer types are supported by LCP: APP_DUAL_TIM_ID_0, APP_DUAL_TIM_ID_1, APP_TIM_ID_0, and APP_TIM_ID_1.

---

### 3.1.1 LCP_TRX_MODE_SW_TX

In this mode, when the application calls the packet TX API gdx_lcp_data_tx, the data packet will be sent as soon as possible.

### 3.1.2 LCP_TRX_MODE_SW_RX

In this mode, when the application calls the packet RX API gdx_lcp_rx_start, packet receiving will start as soon as possible.

### 3.1.3 LCP_TRX_MODE_TIMER_TX

In this mode, when the application calls the packet TX API gdx_lcp_data_tx, packet sending will be triggered at the specified timer moment; after a specified interval, the TX mode automatically switches to RX mode.

### 3.1.4 LCP_TRX_MODE_TIMER_RX

In this mode, when the application calls the packet RX API gdx_lcp_rx_start, packet receiving will be triggered at the specified timer moment; after a specified interval, the RX mode automatically switches to TX mode.

## 3.2 Communication Roles

According to the communication modes in LCP applications, devices can be classified into the following two roles:

- PTX: a receiver that supports LCP_TRX_MODE_SW_TX and LCP_TRX_MODE_TIMER_TX modes
- PRX: a transmitter that supports LCP_TRX_MODE_SW_RX and LCP_TRX_MODE_TIMER_RX modes

### 3.2.1 PTX Timing

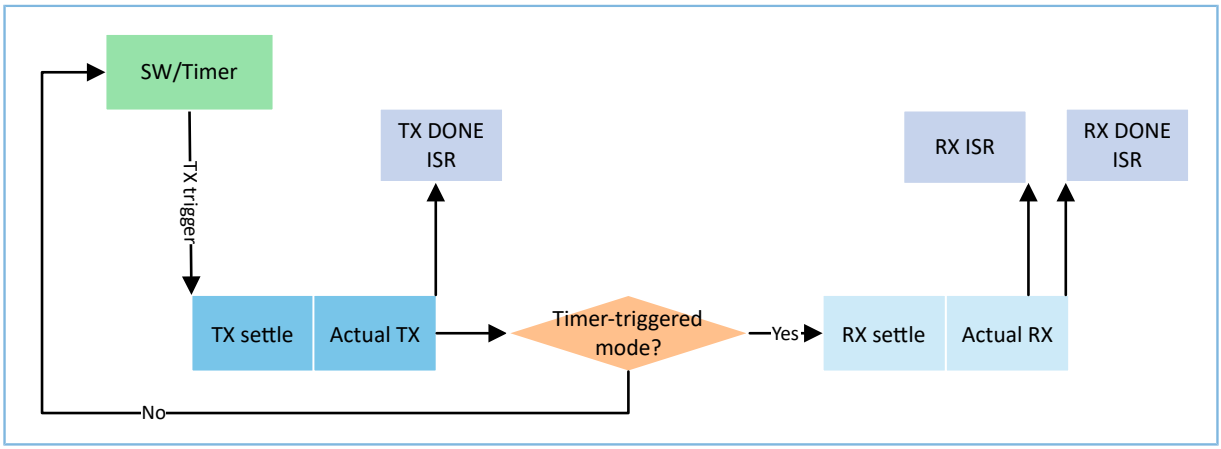


Figure 3-1 PTX timing

**Note**:

Only the timer-triggered TX mode supports automatic switching from TX to RX (T2R), and this function is enabled by default.

### 3.2.2 PRX Timing

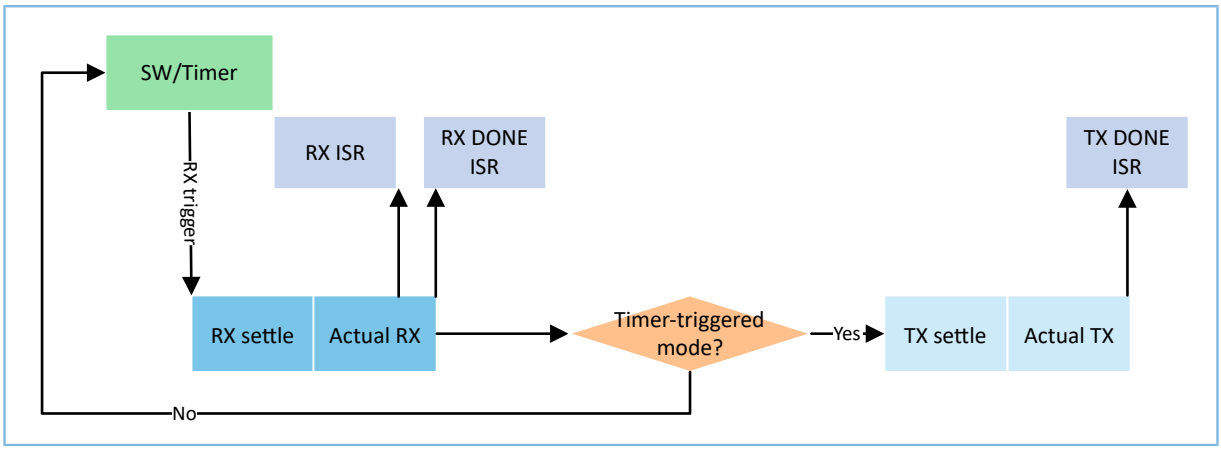


Figure 3-2 PRX timing

**Note**:

Only the timer-triggered RX mode supports automatic switching from RX to TX (R2T), and this function is enabled by default.

## 3.3 LCP APIs

GR533x SDK provides rich and varied LCP APIs, making it convenient for users to utilize LCP.

**Tip**:

For more information about LCP APIs, refer to SDK_Foler\components\sdk\ble_lcp.h; SDK_Foler is the root directory of GR533x SDK.

## 3.3.1 Structure

### 3.3.1.1 gdx_lcp_config_t

The gdx_lcp_config_t structure is used to define the environment variables for LCP communication, such as TX/RX mode, access mode, callback registration, and mode.

- Definition

```
typedef struct {
uint8_t  trx_mode;
int8_t  txpwr_dbm;
uint32_t freq_mhz;
uint32_t access_address;
uint32_t crc_init;
uint32_t rx_window_size_us;
uint8_t  rate;
bool  whiten_en;
bool  b_disable_rx_oneshot_mode;
uint32_t trx_timer_period_us;
uint32_t trx_timer_trigger_trx_time_us;
tx_done_cb_t tx_done_cb;
rx_done_cb_t rx_done_cb;
rx_handler_cb_t  rx_handler_cb;
} gdx_lcp_config_t;
```

- Members

Table 3-1 Members of gdx_lcp_config_t

| Member | Type | Description |
|---|---|---|
| trx_mode | uint8_t | LCP TX/RX mode; refer to “Section 3.3.2.1 PROTOCOL_MODE” for definition. |
| txpwr_dbm | int8_t | TX power; range: –20 dBm to 4 dBm |
| freq_mhz | uint32_t | Operating frequency; value: even frequencies within 2360 MHz–2520 MHz |
| access_address | uint32_t | Device address |
| crc_init | uint32_t | Initial value of CRC<br>0: Disable CRC; no CRC will be performed |
| rx_window_size_us | uint32_t | RX window width; range: 0–10239687<br>0: Enable infinite window. |
| rate | uint8_t | LCP transfer rate; options: 1 Mbps and 2 Mbps |

| Member | Type | Description |
|---|---|---|
| whiten_en | bool | Whether to enable whitening. |
| b_disable_rx_oneshot_mode | bool | Whether to disable one-shot RX mode (only one-shot RX mode is supported in timer-triggered mode).<br>◦ true: Disable one-shot RX mode (use the continuous RX mode). When the RX duration exceeds rx_window_size, exit RX mode.<br>◦ false: Enable one-shot RX mode (exit RX mode after one packet is received). |
| trx_timer_period_us | uint32_t | TX/RX period in timer-triggered mode |
| time_before_trx_timer_expired_us | uint32_t | In timer-triggered mode, the period before the timer triggers a TX or RX operation<br>**Note:**<br>The period shall be shorter than the timer cycle. |
| tx_done_cb | tx_done_cb_t | TX complete interrupt callback function |
| rx_done_cb | rx_done_cb_t | RX complete interrupt callback function |
| rx_handler_cb | rx_handler_cb_t | RX handling interrupt callback function |

## 3.3.2 Enumerations

### 3.3.2.1 PROTOCOL_MODE

The PROTOCOL_MODE enumeration is used to define LCP TX/RX mode.

- Definition

```
enum PROTOCOL_MODE {
    LCP_TRX_MODE_SW_TX = 2,      /*TX by software, just once */
    LCP_TRX_MODE_SW_RX = 3,        /*RX by software, just once */
    LCP_TRX_MODE_TIMER_TX = 4,      /*TX by timer with specified period, will stop when the
 timer stop */
    LCP_TRX_MODE_TIMER_RX = 5,        /*RX by timer with specified period, will stop when the
 timer stop */
};
```

- Members

Table 3-2 Members of PROTOCOL_MODE

| Member | Value | Description |
|---|---|---|
| LCP_TRX_MODE_SW_TX | 2 | One-shot software-triggered TX mode |
| LCP_TRX_MODE_SW_RX | 3 | One-shot software-triggered RX mode |
| LCP_TRX_MODE_TIMER_TX | 4 | Periodic timer-triggered TX mode |
| LCP_TRX_MODE_TIMER_RX | 5 | Periodic timer-triggered RX mode |

#### 3.3.2.2 LCP_RATE

The LCP_RATE enumeration is used to define LCP transfer rate.

- Definition

```
enum LCP_RATE {
    LCP_RATE_1MBPS = 0,
    LCP_RATE_2MBPS,
};
```

- Members

Table 3-3 Members of LCP_RATE

| Member | Value | Description |
|---|---|---|
| LCP_RATE_1MBPS | 0 | Transfer rate: 1 Mbps |
| LCP_RATE_2MBPS | 1 | Transfer rate: 2 Mbps |

### 3.3.3 APIs

#### 3.3.3.1 gdx_lcp_init

Table 3-4 gdx_lcp_init API

| | |
|---|---|
| **Function Prototype** | uint16_t gdx_lcp_init(gdx_lcp_config_t *gdx_lcp_config) |
| **Function Description** | Initialize the LCP module. |
| **Parameter** | gdx_lcp_config: pointer to gdx_lcp_config_t, which contains the environment variables for LCP communication, including TX/RX mode, transfer rate, frequency, and other configuration information |
| **Return Value** | • 0: Succeeded<br>• Others: Failed |
| **Remarks** | This API shall be called after initialization of Bluetooth LE Protocol Stack. |

#### 3.3.3.2 gdx_lcp_deinit

Table 3-5 gdx_lcp_deinit API

| | |
|---|---|
| **Function Prototype** | void gdx_lcp_deinit(void) |
| **Function Description** | Deinitialize the LCP module. |
| **Parameter** | None |
| **Return Value** | None |
| **Remarks** | None |

#### 3.3.3.3 gdx_lcp_timer_binding

Table 3-6 gdx_lcp_timer_binding API

| | |
|---|---|
| **Function Prototype** | void gdx_lcp_timer_binding(bool b_dual_timer, uint8_t timer_id) |
| **Function Description** | Bind the LCP communication behavior with the timer to achieve automatic triggering of TX and RX behaviors by the timer. |
| **Parameter** | • b_dual_timer: Specify the timer type.<br>◦ true: app dual timer<br>◦ false: app timer<br>• timer_id: 0 or 1 |
| **Return Value** | None |
| **Remarks** | • Four timer types are supported: APP_DUAL_TIM_ID_0, APP_DUAL_TIM_ID_1, APP_TIM_ID_0, and APP_TIM_ID_1.<br>• This API shall be called only when timer-triggered TX/RX mode is adopted. |

### 3.3.3.4 gdx_lcp_tx_power_set

Table 3-7 gdx_lcp_tx_power_set API

| | |
|---|---|
| **Function Prototype** | uint16_t gdx_lcp_tx_power_set(int8_t txpwr_dbm) |
| **Function Description** | Set the TX power of data packets. |
| **Parameter** | txpwr_dbm: TX power; range: –20 dBm to 4 dBm |
| **Return Value** | • 0: Succeeded<br>• Others: Failed |
| **Remarks** | |

### 3.3.3.5 gdx_lcp_tx_power_get

Table 3-8 gdx_lcp_tx_power_get API

| | |
|---|---|
| **Function Prototype** | uint16_t gdx_lcp_tx_power_get(int8_t *txpwr_dbm) |
| **Function Description** | Obtain the TX power of data packets. |
| **Parameter** | *txpwr_dbm: pointer to TX power |
| **Return Value** | • 0: Succeeded<br>• Others: Failed |
| **Remarks** | |

### 3.3.3.6 gdx_lcp_channel_set

Table 3-9 gdx_lcp_channel_set API

| | |
|---|---|
| **Function Prototype** | uint16_t gdx_lcp_channel_set(uint32_t freq_mhz) |

| | |
|---|---|
| **Function Description** | Set the operating frequency. |
| **Parameter** | freq_mhz: even frequencies within 2360 MHz–2520 MHz |
| **Return Value** | • 0: Succeeded<br>• Others: Failed |
| **Remarks** | This API shall be called after initialization of the LCP module. |

### 3.3.3.7 gdx_lcp_channel_get

Table 3-10 gdx_lcp_channel_get API

| | |
|---|---|
| **Function Prototype** | uint16_t gdx_lcp_channel_get(uint32_t *freq_mhz) |
| **Function Description** | Obtain the operating frequency. |
| **Parameter** | freq_mhz: pointer to the operating frequency |
| **Return Value** | • 0: Succeeded<br>• Others: Failed |
| **Remarks** | |

### 3.3.3.8 gdx_lcp_data_tx

Table 3-11 gdx_lcp_data_tx API

| | |
|---|---|
| **Function Prototype** | uint16_t gdx_lcp_data_tx(uint8_t header, uint8_t length, uint8_t *p_payload) |
| **Function Description** | Set the data to be sent. |
| **Parameter** | • header: header information contained in the LCP data packet<br>• length: length of the data to be sent; corresponding to “PDU len” in the LCP data packet<br>• p_payload: the data to be sent; corresponding to “PDU” in the LCP data packet |
| **Return Value** | • 0: Succeeded<br>• Others: Failed |
| **Remarks** | • In software-triggered TX mode, data will be sent immediately after this API is called.<br>• In timer-triggered TX mode, after this API is called, data will be sent only after the timer triggers the sending behavior. |

### 3.3.3.9 gdx_lcp_auto_txrx_mode_set

Table 3-12 gdx_lcp_auto_txrx_mode_set API

| | |
|---|---|
| **Function Prototype** | void gdx_lcp_auto_txrx_mode_set(bool enable) |
| **Function Description** | Set the automatic TX/RX switching mode. |
| **Parameter** | enable:<br>• false: Disable automatic TX/RX switching (use the software-triggered RX/TX mode).<br>• true: Enable automatic TX/RX switching (use the timer-triggered RX/TX mode). |

| | |
|---|---|
| **Return Value** | None |
| **Remarks** | |

### 3.3.3.10 gdx_lcp_rx_oneshot_mode_set

Table 3-13 gdx_lcp_rx_oneshot_mode_sett API

| | |
|---|---|
| **Function Prototype** | void gdx_lcp_rx_oneshot_mode_set(bool enable) |
| **Function Description** | Set the RX mode. |
| **Parameter** | enable:<br>• false: Disable one-shot RX mode (use the continuous RX mode). When the RX duration exceeds rx_window_size, exit RX mode.<br>• true: Enable one-shot RX mode (exit RX mode after one packet is received). |
| **Return Value** | None |
| **Remarks** | Only the one-shot RX mode is supported in timer-triggered mode. |

### 3.3.3.11 gdx_lcp_rx_window_size_set

Table 3-14 gdx_lcp_rx_window_size_set API

| | |
|---|---|
| **Function Prototype** | void gdx_lcp_rx_window_size_set(uint32_t time_us) |
| **Function Description** | Set the RX window width. |
| **Parameter** | time_us: range: 0–10239687; 0: Indicate that the RX window width is infinite. |
| **Return Value** | None |
| **Remarks** | |

### 3.3.3.12 gdx_lcp_rx_start

Table 3-15 gdx_lcp_rx_start API

| | |
|---|---|
| **Function Prototype** | uint16_t gdx_lcp_rx_start(void) |
| **Function Description** | Start receiving data packets. |
| **Parameter** | None |
| **Return Value** | • 0: Succeeded<br>• Others: Failed because the previous RX operation not yet completed |
| **Remarks** | • In software-triggered RX mode, data receiving will be started immediately after this API is called.<br>• In timer-triggered RX mode, after this API is called, data receiving will be started only after the timer triggers the RX mechanism. |

### 3.3.3.13 gdx_lcp_rx_stop

Table 3-16 gdx_lcp_rx_stop API

| | |
|---|---|
| **Function Prototype** | void gdx_lcp_rx_stop(void) |
| **Function Description** | Stop receiving data packets. |
| **Parameter** | None |
| **Return Value** | None |
| **Remarks** | |

### 3.3.3.14 gdx_lcp_whitening_seed_set

Table 3-17 gdx_lcp_whitening_seed_set API

| | |
|---|---|
| **Function Prototype** | uint16_t gdx_lcp_whitening_seed_set(uint8_t whitening_seed) |
| **Function Description** | Set the whitening seed for the current communication. |
| **Parameter** | whitening_seed: whitening seed |
| **Return Value** | None |
| **Remarks** | By default, the standard Bluetooth LE whitening algorithm is adopted. For details, refer to “Vol 6/Part B/3.2 Data Whitening” in *Bluetooth Core Spec v5.3*.<br>This algorithm is associated with the Channel Number. Relationship between Channel Number in the SDK and Freq_mhz:<br>Channel Number = (Freq_mhz – 2360)/2 |

### 3.3.3.15 gdx_lcp_t2r_turn_around_time_adjust

Table 3-18 gdx_lcp_t2r_turn_around_time_adjust API

| | |
|---|---|
| **Function Prototype** | void gdx_lcp_t2r_turn_around_time_adjust(uint8_t added_us) |
| **Function Description** | Adjust the time interval for switching from TX to RX; default: 48 μs. |
| **Parameter** | added_us: time to be added to the default |
| **Return Value** | None |
| **Remarks** | This API shall be called only in timer-triggered TX mode. |

### 3.3.3.16 gdx_lcp_r2t_turn_around_time_adjust

Table 3-19 gdx_lcp_r2t_turn_around_time_adjust API

| | |
|---|---|
| **Function Prototype** | void gdx_lcp_r2t_turn_around_time_adjust(uint8_t added_us) |
| **Function Description** | Adjust the time interval for switching from RX to TX; default: 52 μs. |
| **Parameter** | added_us: time to be added to the default |
| **Return Value** | None |

| Remarks | This API shall be called only in timer-triggered RX mode. |
|---|---|

## 3.4 Creating an LCP Application

This section details the implementation procedures for various LCP applications, to help users quickly customize LCP applications.

### 3.4.1 Application in Software-triggered TX/RX Mode

Follow the procedures below to create an application in software-triggered TX/RX mode.

1. Initialize the LCP module by calling gdx_lcp_init().
2. Call gdx_lcp_data_tx() to transmit data packets; call gdx_lcp_rx_start() to start receiving data packets.
3. Call rx_handler_cb() to process the received data.
4. After data transmission/reception, call tx_done_cb/rx_done_cb() to prepare for the next communication.

### 3.4.2 Application in Timer-triggered TX/RX Mode

Follow the procedures below to create an application in timer-triggered TX/RX mode.

1. Initialize the timer.
2. Initialize the LCP module by calling gdx_lcp_init().
3. Call gdx_lcp_timer_binding() to bind the LCP communication behavior to the specified timer.
4. Call gdx_lcp_data_tx() to transmit data packets; call gdx_lcp_rx_start() to start receiving data packets.
5. Call rx_handler_cb() to process the received data.
6. After data transmission/reception, call tx_done_cb/rx_done_cb() to prepare for the next communication.

### 3.4.3 Pulse Detection Example Project

- Function description:

  Input square wave at APP_IO_PIN_0 on the PTX side, and output recovery waveform at APP_IO_PIN_0 on the PRX side, with a recovery accuracy < 0.5 μs without any packet loss over the air.

- Reference example projects:
  - PTX role example project: SDK_Folder\projects\ble\ble_peripheral\ble_app_lcp
  - PRX role example project: SDK_Folder\projects\ble\ble_central\ble_app_lcp_c