Introduction

One of the highlighted new features in Bluetooth 5 is the increased throughput. This is achieved through the 2 Mbps physical layer. In this module we will learn about the *Bluetooth*® low energy (BLE) 2 Mbps physical layer (PHY), how to maintain a connection when using the LE 2M PHY and what the benefits of using the LE 2M PHY are.

In this lab we will be working with the example project *Project Zero* and BTool in order to form a BLE connection and issue a PHY change request. This SimpleLink Academy lab uses projects from the BLE5-Stack, which is found in the SimpleLink™ CC26X2 software development kit (SDK) and the SimpleLink™ CC13X2 software development kit (SDK).

This tutorial will take about two hours to complete and requires basic embedded programming skills (see the **Prerequisites** section for details).

- In the first task, BTool will be used to initiate a connection and send a Read Current PHY, `HCI_LE_ReadPhy` , command from the central device using BTool.
- In the second task, we will try to send a change PHY request from the central device using BTool.
- In the third task, we will try to send a change PHY request from the peripheral device instead.
- In the fourth task, we are going to show what happens when a change PHY request is sent to a device that does not support LE 2M PHY.
- In the first bonus task, we will use the LE Coded PHY.
- In the second bonus task, the LE 2M PHY will be used for advertisements.

> **Attention**
>
> This lab can not be run from CCS Cloud. In order to do this lab, you need to install the relevant SimpleLink™ software development kit locally.

Prerequisites

Other SimpleLink Academy Labs

- Completion of BLE Scanning and Advertising (../ble_scan_adv_basic/ble_scan_adv_basic.html)
- Completion of BLE Connections (../ble_connections/ble_connections.html)

Software

- CCS 8.1+ (Downloadable here (http://www.ti.com/tool/CCSTUDIO)) or IAR 8.20.2+
- BTool (located in the **tools** → **ble5stack** folder of the SimpleLink CC26x2R SDK)

- SimpleLink™ CC26X2 SDK (http://www.ti.com/tool/SIMPLELINK-CC26X2-SDK (http://www.ti.com/tool/SIMPLELINK-CC26X2-SDK))

  or

- SimpleLink™ CC13X2 SDK (http://www.ti.com/tool/SIMPLELINK-CC13X2-SDK (http://www.ti.com/tool/SIMPLELINK-CC13X2-SDK))

Hardware

- 2x CC26X2R1-LAUNCHXL (http://www.ti.com/tool/LAUNCHXL-CC26X2R1)

  or

- 2x CC1352R1-LAUNCHXL (http://www.ti.com/tool/LAUNCHXL-CC1352R1)

Setup

To start with this training module, two LaunchPads will be programmed with

1. *Project Zero* to act as a peripheral device
2. *Host Test* to act as a central device.

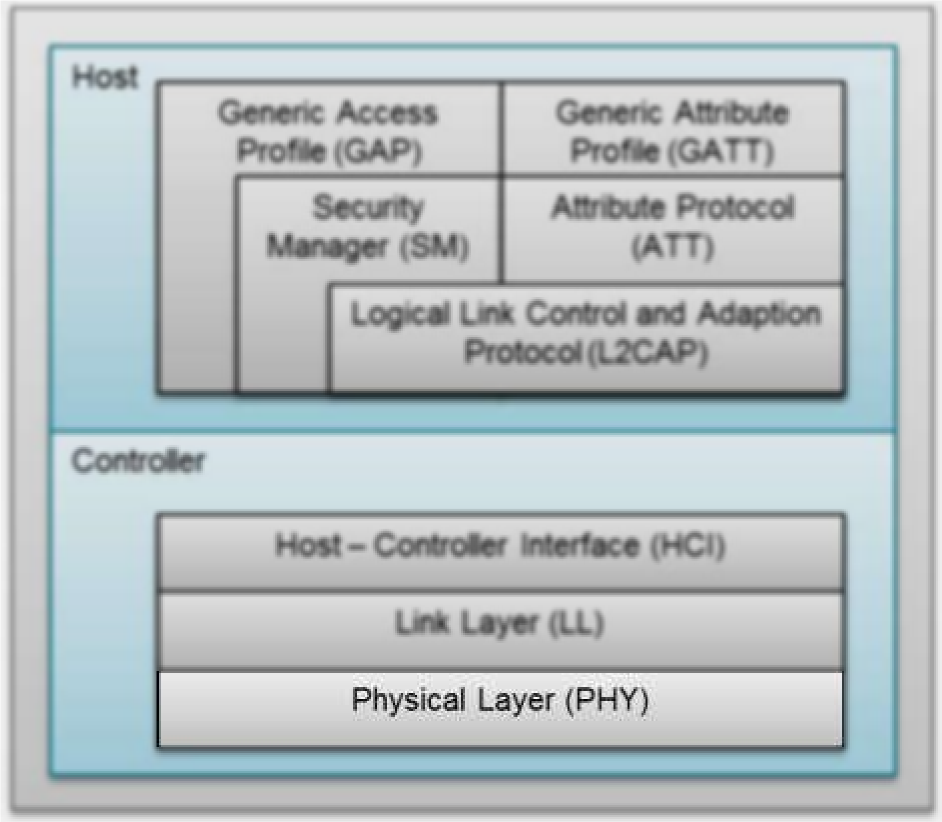
Both example projects can be found in the examples folder in the SDK.

You can use either the CC26X2R1 LaunchPads or CC1352 LaunchPads.

About the LE 2M PHY

The physical layer (PHY) is the lowest layer of the *Bluetooth*® low energy protocol stack. It configures the physical parameters of the radio transmission and reception. It determines how a bit (and its value) are represented over the air.

A full guide of the physical layer is given in the TI BLE5-Stack User's Guide (http://dev.ti.com/tirex/#/?link=Software%2FSimpleLink%20CC26X2%20SDK%2FDocuments%2FBLE5-Stack%2FBLE5-Stack%20User%27s%20Guide), see the PHY section located in **Developing a Bluetooth Low Energy Application** → **The Stack** → **Physical Layer (PHY)**.



One of the key parameters of a physical layer is the **symbol rate**. The symbol rate represents how many symbols are sent/received per second. In this case one symbol represents one bit . When we switch from the 1 Mbps physical layer to the 2 Mbps physical layer, the symbol rate is doubled. This means that data is transmitted at twice the speed. The channel width is the same for LE 1M PHY and LE 2M PHY in order for them to coexist .

The main advantage of the 2 Mbps physical layer over the 1 Mbps physical layer is that the transmission speed is increased. This means that the radio spends less time transmitting/receiving data. This will reduce the device energy consumption for sending the same amount of data.

The primary disadvantage to using the 2 Mbps physical layer is that the radio sensitivity is lowered. This can give a reduced range or a higher packet error rate. A comparison of the 1 Mbps and 2 Mbps physical layers is given in the following table.

Parameter	**Comparison**
Power consumption	For the same transmit power, the energy consumption is reduced when using LE 2M PHY.
Data rate	The LE 2M PHY is two times faster to transmit data than the LE 1M PHY.
Receive sensitivity	The link budget will be lower in the LE 2M PHY relative to the LE 1M PHY, due to the increased symbol rate.
Transmit power	The output power is same for both PHYs.

Quiz

Which of the following use-cases would benefit from using the LE 2M PHY?

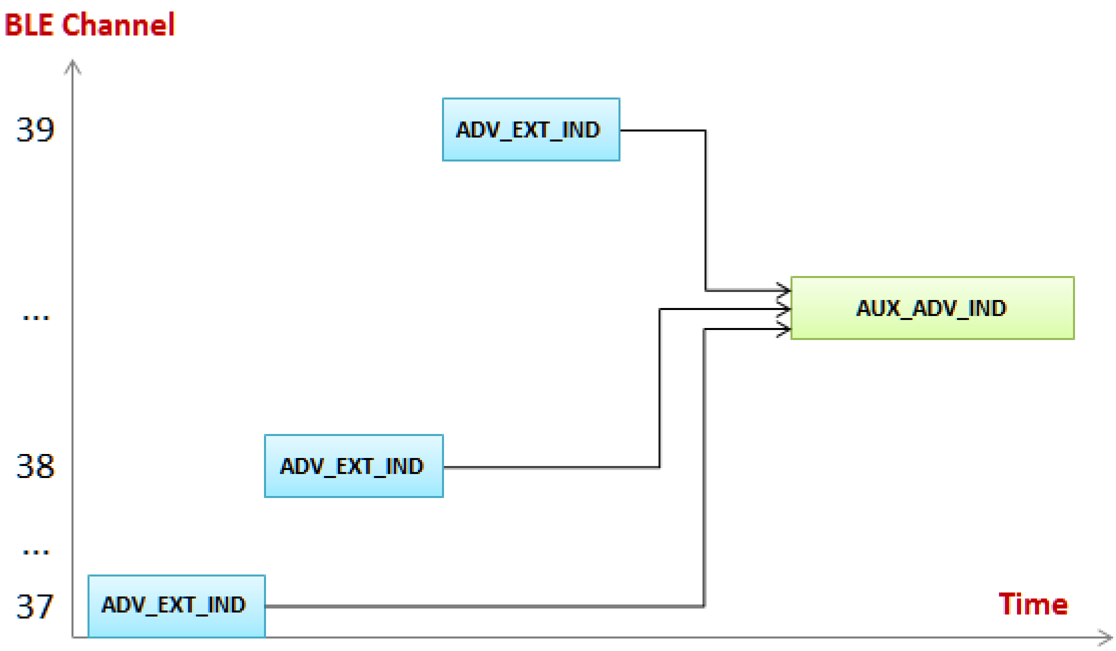
- A use-case where a lot of data must be transmitted
- A use-case in a noisy environment
- A use-case where low latency is important

Bluetooth 5 PHY Possibilities and Restrictions

In the SLA module BLE Scanning and Advertising (../ble_scan_adv_basic/ble_scan_adv_basic.html), we learned that it is possible to advertise on all the Bluetooth low energy PHYs:

- LE 1M PHY
- LE 2M PHY
- LE Coded PHY

According to the Bluetooth Core Spec v.5, only `AUX_ADV_IND`, `AUX_SYNC_IND` and `AUX_CHAIN_IND` can be sent on the LE 2M PHY. All these packets have to be preceded by an `ADV_EXT_IND` packet sent on the LE 1M PHY. The `ADV_EXT_IND` packets contain pointers to the extended advertising packet sent on the secondary PHY.



By using this method, a lot more advertising data can be sent than if you're only using Bluetooth 4.2 advertising (legacy advertising). (254 bytes per packets vs. 31 bytes per packet.) The drawback is that the overhead is increased; you have to transmit multiple packets, and the auxiliary advertisement packets contain the extended advertising header in addition to the advertisement data.

Note that there is no way of transmitting an advertisement packet on the LE 2M PHY without first transmitting a `ADV_EXT_IND` packet on the LE 1M PHY.

In this SimpleLink Academy module we will change the PHY inside a connection. In Bonus Task 2 we will advertise on the LE 2M PHY.

PHY Change in a the Bluetooth Connection

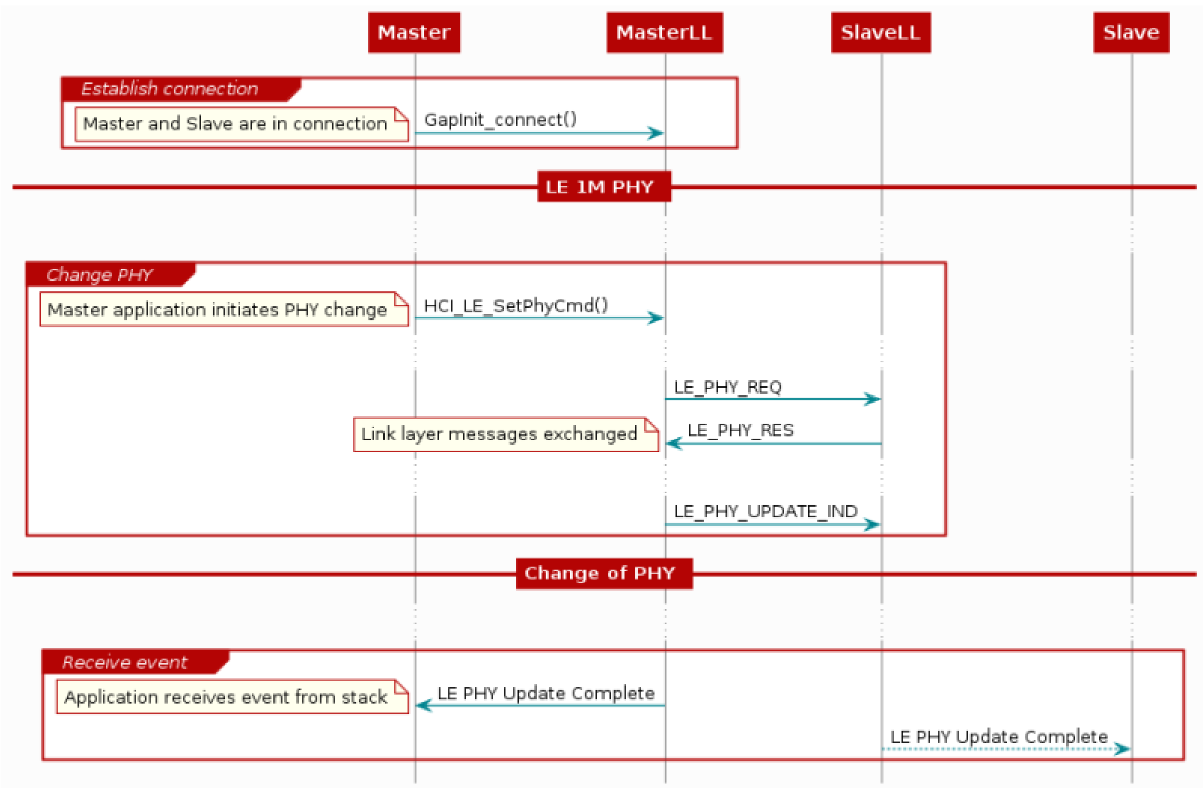
Since the key parameters of the radio changes when we change the PHY of a connection, it is crucial that both devices in the connection agree on the PHY change. This is handled in the BLE5-Stack with a PHY change request.

The *set PHY command* is defined in the Bluetooth Core Spec as `HCI_LE_Set_PHY` .

- You can read about the `HCI_LE_Set_PHY` command in the Bluetooth 5 Core Spec (https://www.bluetooth.com/specifications/bluetooth-core-specification), Vol 2, Part E, section 7.8.49 *LE Set PHY Command*.

In the Bluetooth Core Spec, all the `HCI_LE_Set_PHY` command parameters are described. You can also see what events are generated by this command.

A **HCI** command is a command sent from the **Host** to the **Controller** 🎓 . In this case, the application sends the command to the **Link Layer** 🎓 . In the following figure you can see the `HCI_LE_SetPhyCmd` , and the following `LL_PHY_REQ` and `LL_PHY_RSP` exchange over the air.



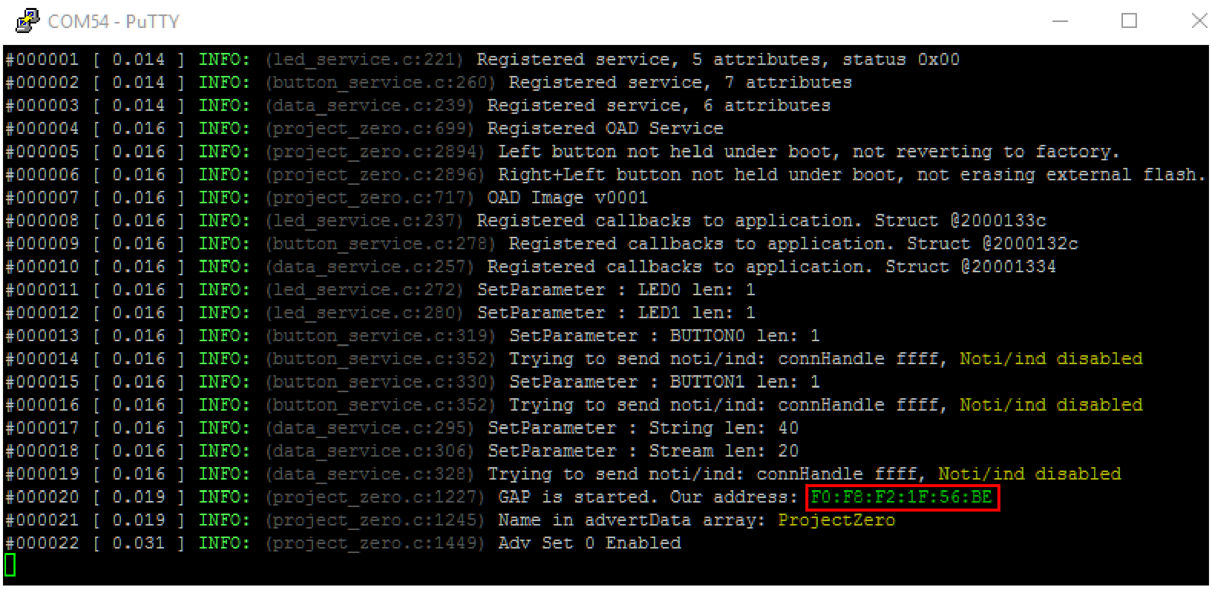
Both the master and the slave device are given the chance to refuse the PHY change. In the `LL_PHY_REQ` and `LL_PHY_RSP` , each device indicates which PHYs they prefer to use. The PHY will not be updated unless both devices have named the PHY as a preferred

PHY.

Task 1 – Initiate a Connection and Read the Current PHY

Program one LaunchPad with *Project Zero*, and one with *Host Test* from BLE5-Stack. (If you don't remember how to do this, a description is given in BLE Connections (../ble_connections/ble_connections.html).) Open a UART logging window for Project Zero.

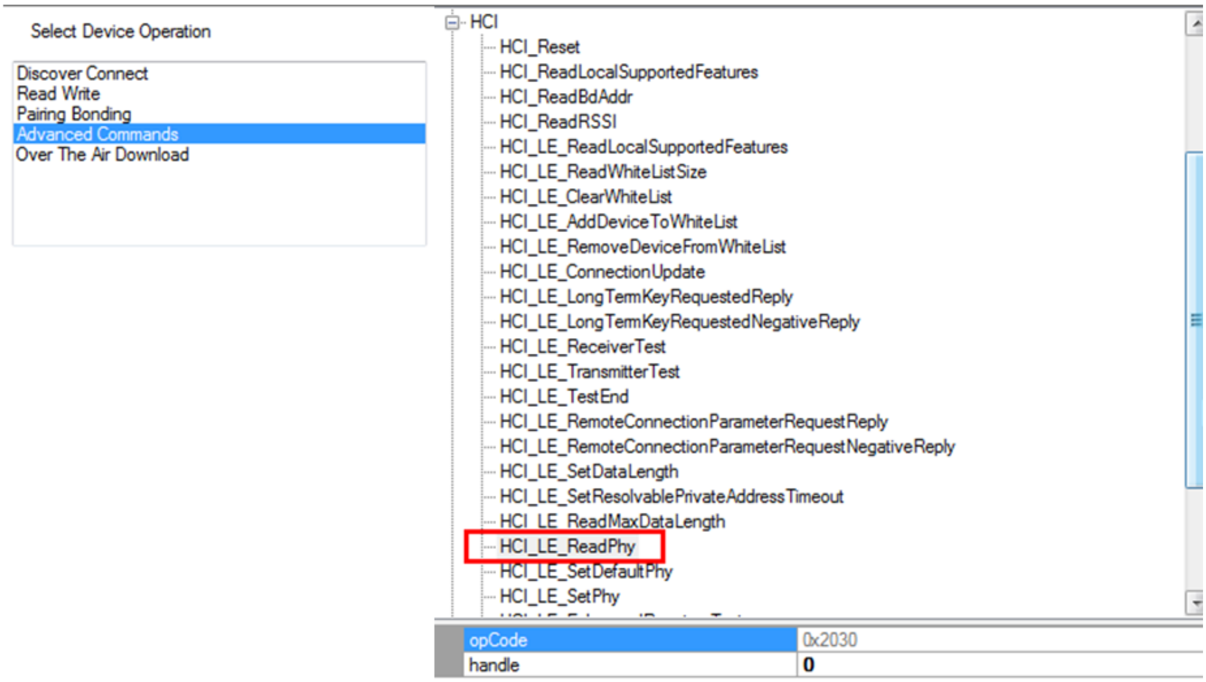
Open BTool by using the 'Run-BTool.bat' file. In order to connect with Project Zero from BTool, we need the peripheral's device address BDA . The peripheral's device address can be found in the UART log as shown below:



In BTool, press the **Scan** button. BTool will start printing out `GAP_AdvertiserScannerEvent` logs. Connect to Project Zero by using its BDA.

Now, that a link is established, we are going to try to figure out which PHY is currently used in this connection. The command we will use is `HCI_LE_ReadPhy` , which can be found under Advanced Commands in BTool.

- You can read more about the `HCI_LE_ReadPhy` command in the Bluetooth 5 Core Spec (https://www.bluetooth.com/specifications/bluetooth-core-specification), Vol 2, Part E, 7.8.47 *LE Read PHY Command*.



The `HCI_LE_ReadPhy` command is used to read the current transmitter PHY and receiver PHY on the connection identified by the (connection) handle.

Once the `HCI_LE_ReadPhy` command is issued, the device which issued the command will generate a `HCI_CommandCompleteEvent` 🎓 . The return event contains the following fields:

```
------------------------------------------------------------------
[43] : <Tx> - 02:59:38.028
-Type           : 0x01 (Command)
-OpCode         : 0x2030 (HCI_LE_ReadPhy)
-Data Length    : 0x02 (2) byte(s)
 Handle         : 0x0000 (0)
Dump(Tx):
0000:01 30 20 02 00 00                              .0 ...
------------------------------------------------------------------
[44] : <Rx> - 02:59:38.058
-Type           : 0x04 (Event)
-EventCode      : 0x000E (HCI_CommandCompleteEvent)
-Data Length    : 0x08 (8) bytes(s)
 Packets        : 0x01 (1)
 OpCode         : 0x2030 (HCI_LE_ReadPhy)
 Status         : 0x00 (0) (Success)
 Handle         : 0x0000 (0)
 TxPhy          : 0x01 (1) (Phy_1_Mbps)
 RxPhy          : 0x01 (1) (Phy_1_Mbps)
Dump(Rx):
0000:04 0E 08 01 30 20 00 00 00 01 01               ....0 .....
------------------------------------------------------------------
```

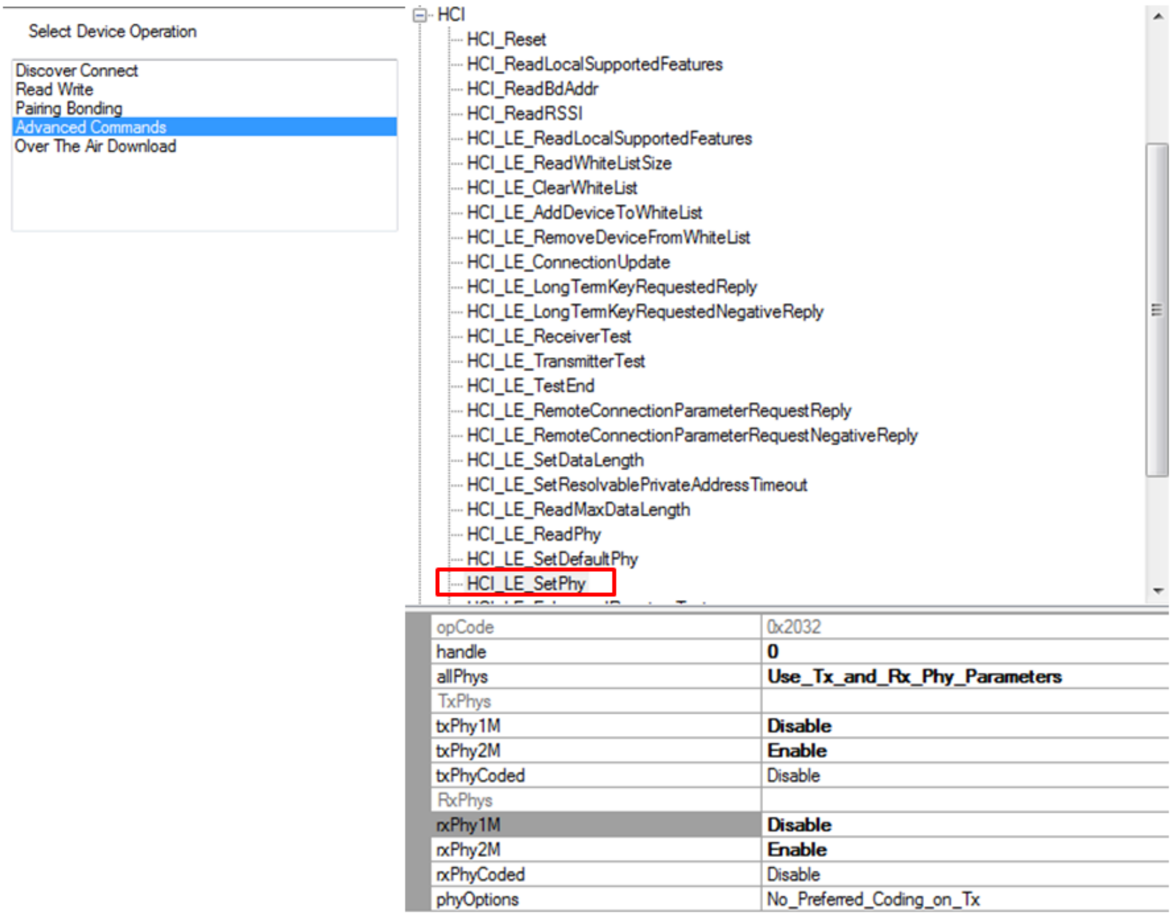
As you can see, the default PHY used to establish the connection is the LE 1M PHY. The table shows how to interpret different return values for the TxPhy and RxPhy fields.

Value	Definition
1	The transmitter/receiver PHY for the connection is LE 1M PHY.
2	The transmitter/receiver PHY for the connection is LE 2M PHY.
3	The transmitter/receiver PHY for the connection is LE Coded PHY.

Task 2 – Send 'Change PHY Request' using BTool.

As described in the PHY section of the TI BLE5-Stack User's Guide (http://dev.ti.com/tirex/#/?link=Software%2FSimpleLink%20CC26X2%20SDK%2FDocuments%2FBLE5-Stack%2FBLE5-Stack%20User%27s%20Guide) (located in **Developing a Bluetooth Low Energy Application** → **The Stack** → **Physical Layer (PHY)**), BLE5 projects in the SimpleLink SDK will by default support both the LE 1M PHY and the LE 2M PHY.

To change the PHY used in a connection, we can use the `HCI_LE_SetPhy` command, which can be found under Adv. Commands in BTool.



In order to use this command, we need to understand all the parameter fields.

- The parameters of `HCI_LE_SetPhy` are listed in the Bluetooth 5 Core Spec (https://www.bluetooth.com/specifications/bluetooth-core-specification), Vol 2, Part E, section 7.8.49 *LE Set PHY Command*.

The following lists all the parameters for `HCI_LE_SetPhy` and explains the parameters.

Parameters for HCI_LE_SetPhy

handle

This is the connection handle. For more information regarding how to identify the current connection handle, please take a look at the SimpleLink Academy training module BLE Connections (../ble_connections/ble_connections.html).

allPhys

The host preference on how to handle txPhy and rxPhy. There are four options for the **allPhys** field:

Option	Definition
Use_Tx_and_Rx_Phy_Parameter	The final preferred PHY setting will be taken from the txPhy and rxPhy fields
Any_Tx_Phy_Use_Rx_Phy_Parameter	The final preferred PHY setting will be taken from the rxPhy field
Use_Tx_Phy_Parameter_Any_Rx_Phy	The final preferred PHY setting will be taken from the txPhy field
Any_Tx_or_Rx_Phy	The sender does not have a preferred PHY. The txPhy/rxPhy fields will be ignored. The current PHY will be updated to the fastest PHY both devices support.

txPhy & rxPhy

Bit field of Host preferred Tx and Rx PHYs. The definition of the **txPhy** and **rxPhy** fields are shown below. In Btool, these bit fields can be toggled by selecting **Enable** or **Disable** for each **txPhy** and **rxPhy** option:

Bit Number	Value	Definition
0	1	The host prefers to use LE 1M PHY as Tx/Rx PHY.
1	2	The host prefers to use LE 2M PHY as Tx/Rx PHY.
2	4	The host prefers to use LE Coded PHY as Tx/Rx PHY.

The TI BLE5-Stack does not support asymmetric connections, i.e. connections where different PHYs are used for Rx and Tx.

phyOptions

Bit field of the Host's preferred Coded PHY options. We are not going to discuss the LE Coded PHY in this training, we will just use `No_Preferred_Coding_on_Tx` for this setting.

Changing PHY

Choose parameters to change to LE 2M PHY. (If you're unsure of what to choose, look at the printout from BTool below.) Send the command. When the command is sent, BTool will output the following:

```
------------------------------------------------------------------
[30] : <Tx> - 02:34:40.788
-Type              : 0x01 (Command)
-OpCode            : 0x2032 (HCI_LE_SetPhy)
-Data Length       : 0x07 (7) byte(s)
 Handle            : 0x0000 (0)
 AllPhys           : 0x00 (0) (Use Tx and Rx Phy Parameters)
 TxPhy             : 0x02 (2) (Phy_2_Mbps)
 RxPhy             : 0x02 (2) (Phy_2_Mbps)
 PhyOptions        : 0x0000 (0) (No Preferred Coding on Tx)
Dump(Tx):
0000:01 32 20 07 00 00 00 02 02 00 00                .2 ........
------------------------------------------------------------------
[31] : <Rx> - 02:34:40.814
-Type              : 0x04 (Event)
-EventCode         : 0x000F (HCI_CommandStatusEvent)
-Data Length       : 0x04 (4) bytes(s)
 Status            : 0x00 (0) (Success)
 Num HCI Cmds      : 0x01 (1)
 OpCode            : 0x2032 (HCI_LE_SetPhy)
Dump(Rx):
0000:04 0F 04 00 01 32 20                            .....2
------------------------------------------------------------------
[32] : <Rx> - 02:34:41.640
-Type              : 0x04 (Event)
-EventCode         : 0x003E (HCI_LE_GenericReportEvent)
-Data Length       : 0x06 (6) bytes(s)
 LE Event Code     : 0x0C (12) (HCI_LE_PhyUpdateCompleteEvent)
 Status            : 0x00 (0) (Success)
 Handle            : 0x0202 (514)
Dump(Rx):
0000:04 3E 06 0C 00 00 00 02 02                      .>.......
------------------------------------------------------------------
```

To verify that we actually changed the PHY, we can use `HCI_LE_ReadPhy` .

```
-----------------------------------------------------------------------
[33] : <Tx> - 02:38:36.090
-Type                : 0x01 (Command)
-OpCode              : 0x2030 (HCI_LE_ReadPhy)
-Data Length         : 0x02 (2) byte(s)
 Handle              : 0x0000 (0)
Dump(Tx):
0000:01 30 20 02 00 00                                  .0 ...
-----------------------------------------------------------------------
[34] : <Rx> - 02:38:36.107
-Type                : 0x04 (Event)
-EventCode           : 0x000E (HCI_CommandCompleteEvent)
-Data Length         : 0x08 (8) bytes(s)
 Packets             : 0x01 (1)
 OpCode              : 0x2030 (HCI_LE_ReadPhy)
 Status              : 0x00 (0) (Success)
 Handle              : 0x0000 (0)
 TxPhy               : 0x02 (2) (Phy_2_Mbps)
 RxPhy               : 0x02 (2) (Phy_2_Mbps)
Dump(Rx):
0000:04 0E 08 01 30 20 00 00 00 02 02                   ....0 .....
-----------------------------------------------------------------------
```

Quiz

What will happen if we set `3` (LE 1M PHY | LE 2M PHY) instead of `2` (LE 2M PHY) for both the txPhy and rxPhy fields in the `HCI_LE_SetPhy` command?

- The connection will alter between LE 1M PHY and LE 2M PHY between each connection event.
- The connection will use the LE 1M PHY.
- The connection will use the LE 2M PHY.
- The connection will be dropped.

Important

When changing the active PHY, the preferred PHY of the PHY change initiating device will also be changed automatically. This means that in some cases, only the the device that first changed the active PHY to LE 2M PHY can change it back to LE 1M PHY. The device that initiated the PHY change to LE 2M will still be able to change the PHY back to LE 1M.

Because of this, it is a good idea to change the PHY to `0x03` (LE 1M PHY | LE 2M PHY) when the application has finished all LE 2M critical operations. This will allow the peer device to change the PHY back to LE 1M.

Task 3 – Change PHY from Project Zero

So far we have learned how to change the PHY using BTool. Now we are moving on to changing PHY from Project Zero by a button press. We will use a button press callback function where the `HCI_LE_SetPhyCmd()` is sent.

1. Send `HCI_LE_SetPhyCmd()`

We will be using `HCI_LE_SetPhyCmd()` in the *Project Zero* code. The parameters of this function are the same as when using the command in BTool, however their values are slightly different.

- Please see `hci.h` for the API description of `HCI_LE_SetPhyCmd()` (you can find it in the *Include* folder of the app project).

```
hciStatus_t HCI_LE_SetPhyCmd( uint16  connHandle,
                              uint8   allPhys,
                              uint8   txPhy,
                              uint8   rxPhy,
                              uint16  phyOpts );
```

hci.h – Definition of `HCI_LE_SetPhyCmd()` .

Use the API reference to choose the correct parameters for `HCI_LE_SetPhyCmd()` . *Project Zero* keeps track of connected devices in the `connList` variable.

```
// Per-handle connection info
static pzConnRec_t connList[MAX_NUM_BLE_CONNS];
```

project_zero.c – The connection list variable is declared in the LOCAL VARIABLES section of the code.

When a device connects to Project Zero, it's added to the `connList` . We can read the connection handle from this list.

```
// Connected device information
typedef struct
{
    uint16_t connHandle;                        // Connection Handle
    Clock_Struct* pUpdateClock;                 // pointer to clock struct
    bool phyCngRq;                              // Set to true if PHY change request is
in progress
    uint8_t currPhy;                            // The active PHY for a connection
    uint8_t rqPhy;                              // The requested PHY for a connection
    uint8_t phyRqFailCnt;                       // PHY change request fail count
} pzConnRec_t;
```

project_zero.c – Typedef for `pzConnRec_t` .

Since we want the PHY to change with a button press, we need to locate the function `ProjectZero_handleButtonPress()` in `project_zero.c` . This is the callback function where button presses are handled.

```
/*********************************************************************
 * @fn      ProjectZero_handleButtonPress
 *
 * @brief   Handle a debounced button press or release in Task context.
 *          Invoked by the taskFxn based on a message received from a callback.
 *
 * @see     buttonDebounceSwiFxn
 * @see     buttonCallbackFxn
 *
 * @param   pState  pointer to pzButtonState_t message sent from debounce Swi.
 *
 * @return  None.
 */
static void ProjectZero_handleButtonPress(pzButtonState_t *pState)
{
    Log_info2("%s %s",
              (uintptr_t)(pState->pinId ==
                          Board_PIN_BUTTON0 ? "Button 0" : "Button 1"),
              (uintptr_t)(pState->state ?
                          ANSI_COLOR(FG_GREEN)"pressed"ANSI_COLOR(ATTR_RESET) :
                          ANSI_COLOR(FG_YELLOW)"released"ANSI_COLOR(ATTR_RESET)
                         ));

    // Update the service with the new value.
    // Will automatically send notification/indication if enabled.
    switch(pState->pinId)
    {
    case Board_PIN_BUTTON0:
        ButtonService_SetParameter(BS_BUTTON0_ID,
                                   sizeof(pState->state),
                                   &pState->state);
        break;
    case Board_PIN_BUTTON1:
        ButtonService_SetParameter(BS_BUTTON1_ID,
                                   sizeof(pState->state),
                                   &pState->state);
        break;
    }
}
```

project_zero.c :: ProjectZero_handleButtonPress() – Project Zero button press handle function.

By default, `ProjectZero_handleButtonPress()` outputs button presses over uart (`Log_info2()`) and updates the button service when a button is pressed or released (`ButtonService_SetParameter()`). We want it to update the PHY only when a button is pressed, not when it is released. When the button is pressed, the `state` member of the `pState` is `1`. Thus we should check this variable before sending a `HCI_LE_SetPhyCmd()`.

Write your code so that when **Button 0** is pressed, the PHY is changed to LE 2M PHY, and when **Button 1** is pressed, the PHY is changed to LE 1M PHY.

ProjectZero_handleButtonPress() Solution

```
/*********************************************************************
* @fn      ProjectZero_handleButtonPress
*
```

```
 * @brief   Handle a debounced button press or release in Task context.
 *          Invoked by the taskFxn based on a message received from a callback.
 *
 * @see     buttonDebounceSwiFxn
 * @see     buttonCallbackFxn
 *
 * @param   pState  pointer to pzButtonState_t message sent from debounce Swi.
 *
 * @return  None.
 */
static void ProjectZero_handleButtonPress(pzButtonState_t *pState)
{
    Log_info2("%s %s",
              (uintptr_t)(pState->pinId ==
                          Board_PIN_BUTTON0 ? "Button 0" : "Button 1"),
              (uintptr_t)(pState->state ?
                          ANSI_COLOR(FG_GREEN)"pressed"ANSI_COLOR(ATTR_RESET) :
                          ANSI_COLOR(FG_YELLOW)"released"ANSI_COLOR(ATTR_RESET)
                         ));

    // Update the service with the new value.
    // Will automatically send notification/indication if enabled.
    switch(pState->pinId)
    {
    case Board_PIN_BUTTON0:
    {
        ButtonService_SetParameter(BS_BUTTON0_ID,
                                   sizeof(pState->state),
                                   &pState->state);
        // Only send setPhy cmd when the button is pressed
        if (pState->state == 1)
        {
              // Set Phy Preference on the current connection. Apply the same v
alue for Rx and Tx.
              HCI_LE_SetPhyCmd(connList[0].connHandle, HCI_PHY_USE_PHY_PARAM, H
CI_PHY_2_MBPS, HCI_PHY_2_MBPS, LL_PHY_OPT_NONE);
              Log_info0("Try to set PHY to 2 Mbps");
        }
    }
    break;
    case Board_PIN_BUTTON1:
    {
        ButtonService_SetParameter(BS_BUTTON1_ID,
                                   sizeof(pState->state),
                                   &pState->state);
        // Only send setPhy cmd when the button is pressed
        if (pState->state == 1)
        {
              // Set Phy Preference on the current connection. Apply the same v
alue for Rx and Tx.
              HCI_LE_SetPhyCmd(connList[0].connHandle, HCI_PHY_USE_PHY_PARAM, H
CI_PHY_1_MBPS, HCI_PHY_1_MBPS, LL_PHY_OPT_NONE);
              Log_info0("Try to set PHY to 1Mbps");
        }
    }
    break;
    }
}
```

project_zero.c :: user_handleButtonPress() – Change the PHY with a button press.

Once the `HCI_LE_SetPhyCmd()` is sent, the controller will post a `hciEvt_BLEPhyUpdateComplete_t` event.

If the `HCI_LE_SetPhyCmd()` is executed successfully, `HCI_LE_SetPhyCmd()` will return `status = SUCCESS`. Then you can check the currently used Tx/Rx PHY.

Program Project Zero, connect it to BTool, and use the **Button 0** to change the PHY to LE 2M PHY.

```
#000084 [ 700.626 ] INFO: (project_zero.c:2658) Button interrupt: Button 0
#000085 [ 700.677 ] INFO: (project_zero.c:2033) Button 0 pressed
#000086 [ 700.677 ] INFO: (button_service.c:319) SetParameter : BUTTON0 len: 1
#000087 [ 700.677 ] INFO: (button_service.c:352) Trying to send noti/ind: connHandle ffff, Noti/ind disabled
#000088 [ 700.677 ] INFO: (project_zero.c:2049) Try to set PHY to 2Mbps
#000089 [ 700.677 ] INFO: (project_zero.c:1386) PHY Update Status Event: 0x0
#000090 [ 700.808 ] INFO: (project_zero.c:2658) Button interrupt: Button 0
#000091 [ 700.858 ] INFO: (project_zero.c:2033) Button 0 released
#000092 [ 700.858 ] INFO: (button_service.c:319) SetParameter : BUTTON0 len: 1
#000093 [ 700.858 ] INFO: (button_service.c:352) Trying to send noti/ind: connHandle ffff, Noti/ind disabled
#000094 [ 701.016 ] INFO: (project_zero.c:1420) PHY Updated to 2M
```

Task 4 – Send a PHY Change Request to a Device that Does Not Support BLE 5

Connect Project Zero to a phone/master device that does not support BLE 5, for example your phone. You have to terminate the connection with BTool in order to do this. Press **Button 0** to request a PHY change to LE 2M PHY. Please note that recently released phones do have BLE 5 support. Do a quick internet search of your phone model's BLE support to check if you can use your phone for this task.

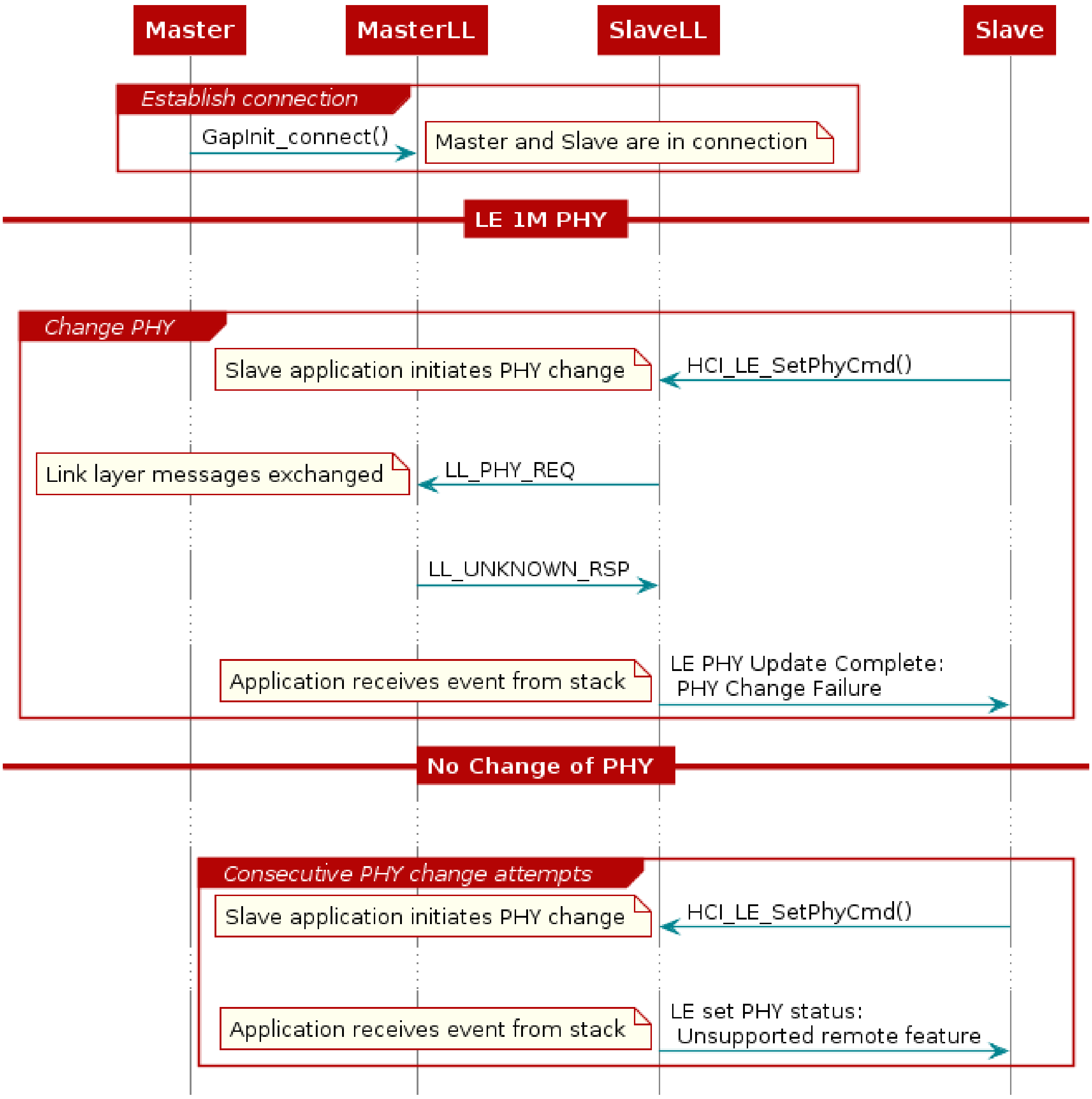
```
#000026 [ 30.550 ] INFO: (project_zero.c:1511) Button interrupt: Button 0
#000027 [ 30.601 ] INFO: (project_zero.c:827) Button 0 pressed
#000028 [ 30.601 ] INFO: (button_service.c:314) SetParameter : BUTTON0 len: 1
#000029 [ 30.601 ] INFO: (button_service.c:345) Trying to send noti/ind: connHandle ffff, Noti/ind disabled
#000030 [ 30.601 ] INFO: (project_zero.c:843) Try to set PHY to 2Mbps
#000031 [ 30.601 ] INFO: (project_zero.c:1137) PHY Update Status Event: 0x0
#000032 [ 30.670 ] INFO: (project_zero.c:1110) PHY Change failure
#000033 [ 30.698 ] INFO: (project_zero.c:1511) Button interrupt: Button 0
#000034 [ 30.748 ] INFO: (project_zero.c:827) Button 0 released
#000035 [ 30.748 ] INFO: (button_service.c:314) SetParameter : BUTTON0 len: 1
#000036 [ 30.748 ] INFO: (button_service.c:345) Trying to send noti/ind: connHandle ffff, Noti/ind disabled
```

Quiz

If the master device doesn't support LE 2M PHY, what happens after sending the `HCI_LE_SetPhyCmd()` command?

- The connection will drop.
- The connection will continue using the LE 1M PHY and the peripheral device will get a callback stating that the PHY change failed.
- The connection will use the LE 2M PHY.
- The phone app will crash.

As you can see from the Project Zero log, the PHY update failed. When the phone (the master in this connection) receives the `LL_PHY_REQ` packet, it does not recognize it since it does not support Bluetooth 5. The phone responds with a `LL_UNKNOWN_RSP` packet. This is illustrated in the following flow chart.



The first time Project Zero attempts changing the PHY and the phone replies with `LL_UNKNOWN_RSP`, the Link Layer notes that the phone does not support PHY change. The next time you try to change to LE 2M PHY, HCI will notify the Project Zero application that this feature is not supported directly. This time, no packets are sent over the air.

If you want to see how the PHY change failures are handled, check out `ProjectZero_processHCIMsg()` in `project_zero.c`.

Bonus Task 1 – Change to Coded PHY

We can use `HCI_LE_SetPhyCmd()` to change the connection PHY to a LE Coded PHY. These PHYs increase the range of the BLE connection.

- If you want to know more about the LE Coded PHYs, and how the range can be increased, please see How does Bluetooth® 5 increase the achievable range of a Bluetooth low energy connection? (https://e2e.ti.com/blogs_/b/connecting_wirelessly/archive/2017/01/30/how-does-bluetooth-5-increase-the-achievable-range-of-a-bluetooth-low-energy-connection)
- You can also read more about LE Coded PHY in the BLE5-Stack User's Guide (http://dev.ti.com/tirex/#/?link=Software%2FSimpleLink%20CC26X2%20SDK%2FDocuments%2FBLE5-Stack%2FBLE5-Stack%20User%27s%20Guide), located in **Developing a Bluetooth Low Energy Application** → **The Stack** → **Physical Layer (PHY)** → **LE Coded PHY**.

You can use the `HCI_LE_SetPhy` command to change to the LE Coded PHY. The bit-mask value for LE Coded PHY is 0x04. You can use either BTool or Project Zero to change to LE Coded PHY.

> **Supported PHYs**
>
> As previously mentioned, if multiple PHYs are supported the PHY with the highest data throughput will be chosen by the controller. Thus, if there are multiple supported PHYs, the LE Coded PHY will not be chosen.

When changing to the LE Coded PHY, you can choose whether you prefer the S=2 coding or the S=8 coding in the `phyOptions` parameter. You can read about the coding options in the BLE5-Stack User's Guide (http://dev.ti.com/tirex/#/?link=Software%2FSimpleLink%20CC26X2%20SDK%2FDocuments%2FBLE5-Stack%2FBLE5-Stack%20User%27s%20Guide) located in **Developing a Bluetooth Low Energy Application** → **The Stack** → **Physical Layer (PHY)** → **LE Coded PHY**.

Bonus Task 2 – Advertise on LE 2M PHY

As previously stated, it is possible to use auxiliary advertisement packets to transmit advertisement data on LE 2M PHY. In order to do this, configure the advertisement set parameters as follows:

```
// Parameters for 2M connectable advertising
#define GAPADV_PARAMS_2M_CONN {                           \
  .eventProps = GAP_ADV_PROP_CONNECTABLE,                 \
  .primIntMin = 160,                                      \
  .primIntMax = 160,                                      \
  .primChanMap = GAP_ADV_CHAN_ALL,                        \
  .peerAddrType = PEER_ADDRTYPE_PUBLIC_OR_PUBLIC_ID,      \
  .peerAddr = { 0xaa, 0xaa, 0xaa, 0xaa, 0xaa, 0xaa },    \
  .filterPolicy = GAP_ADV_WL_POLICY_ANY_REQ,              \
  .txPower = GAP_ADV_TX_POWER_NO_PREFERENCE,              \
  .primPhy = GAP_ADV_PRIM_PHY_1_MBPS,                     \
  .secPhy = GAP_ADV_SEC_PHY_2_MBPS,                       \
  .sid = 0                                                \
}

GapAdv_params_t advParam2M = GAPADV_PARAMS_2M_CONN;

// Create Advertisement set and assign handle
status = GapAdv_create(&ProjectZero_advCallback, &advParam2M, &advHandleLegacy);
APP_ASSERT(status == SUCCESS);
```

project_zero.c :: ProjectZero_processGapMessage() :: GAP_DEVICE_INIT_DONE_EVENT – Configure 2M advertising.

Note the following:

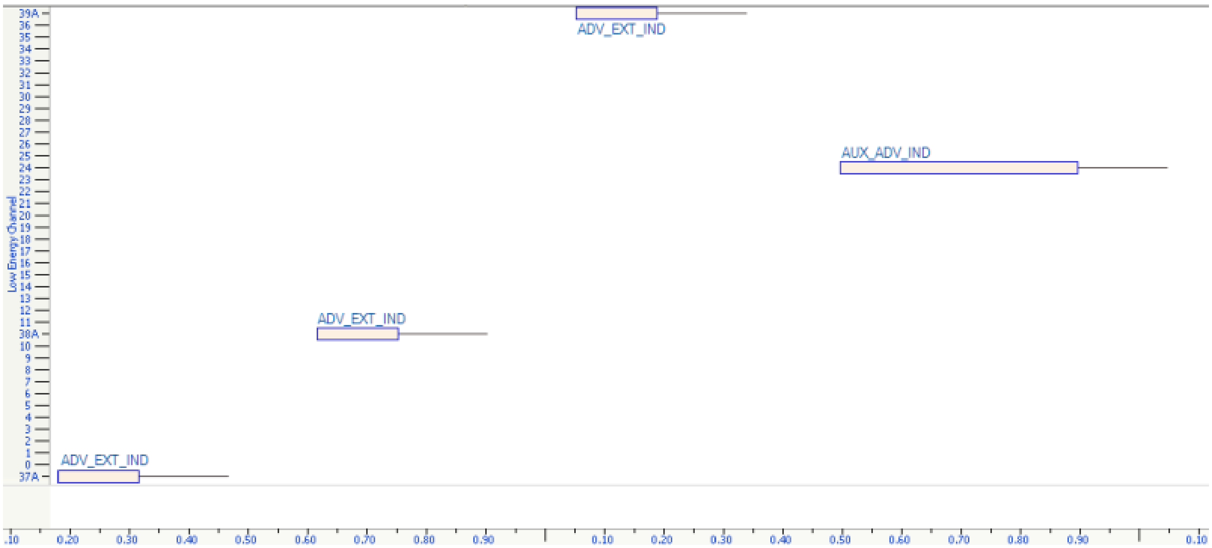
1. The `eventProps` parameter does **not** contain the legacy advertising flag, `GAP_ADV_PROP_LEGACY`
2. Also, advertising on LE 2M PHY can be **either** connectable **or** scannable. In this case it is only connectable (`GAP_ADV_PROP_CONNECTABLE`).
3. The primary PHY (`primPhy`) is the LE 1M PHY. This is the PHY on which we will transmit the `ADV_EXT_IND` packets that contain the pointer to the `AUX_ADV_IND` packet.
4. The secondary PHY (`secPhy`) is, of course, the LE 2M PHY.

You can use BTool to scan for advertisements. The printed `GAP_AdvertiserScannerEvent` will have the LE 2M PHY as the secondary PHY.

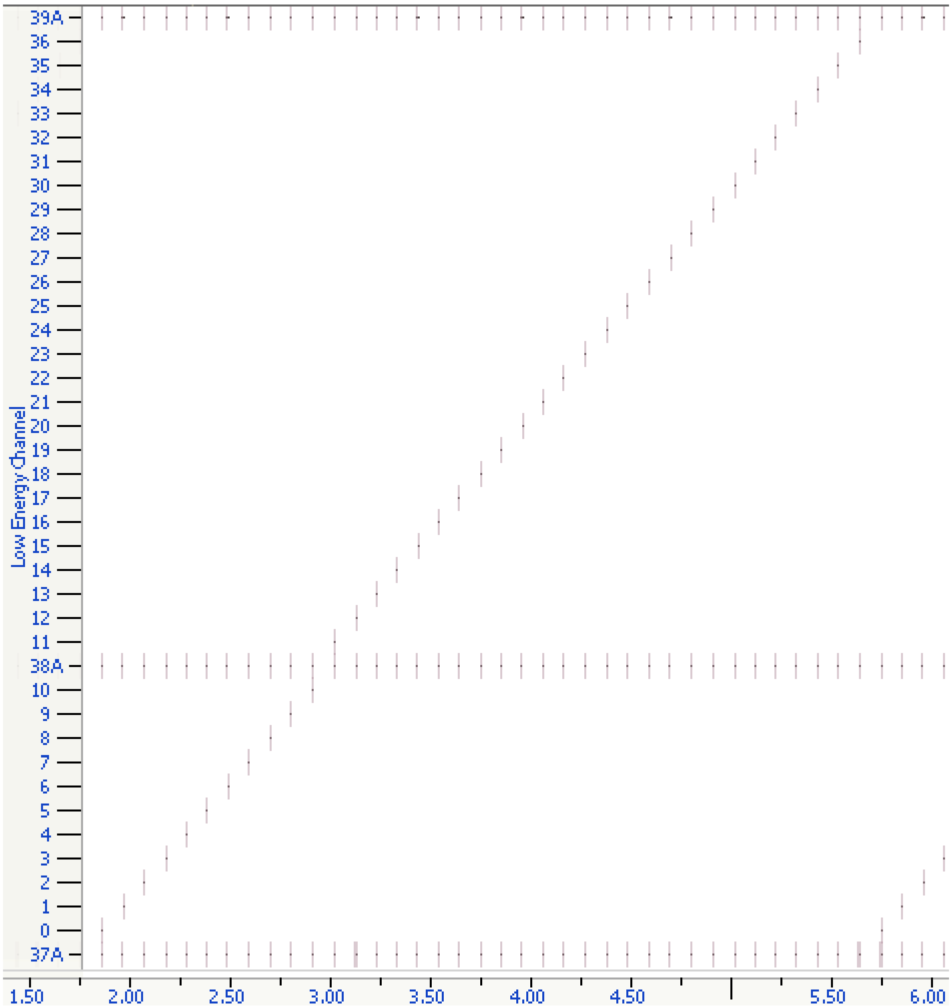
```
------------------------------------------------------------------------
[35] : <Rx> - 09:59:11.262
-Type               : 0x04 (Event)
-EventCode          : 0x00FF (HCI_LE_ExtEvent)
-Data Length        : 0x2F (47) bytes(s)
 Event              : 0x0613 (1555) (GAP_AdvertiserScannerEvent)
 Status             : 0x00 (0) (SUCCESS)
 EventId            : 0x00400000 (4194304) (
                      GAP_EVT_ADV_REPORT)
 AdvRptEventType: 0x01 (1) (
                      AE_Undir_NC_NS_or_Data_Complete
                      AE_Undir_Conn_or_Data_Complete)
 AddressType        : 0x00 (0) (ADDRTYPE_PUBLIC)
 Address            : F0:F8:F2:1F:56:BE
 PrimaryPHY         : 0x01 (1) (SCANNED_PHY_1M)
 SecondaryPHY       : 0x02 (2) (SCANNED_PHY_2M)
 AdvSid             : 0x00 (0)
 TxPower            : 0x7F (127)
 RSSI               : 0xC0 (192)
 DirectAddrType : 0xFF (255) (ADDRTYPE_NONE)
 DirectAddr         : 00:00:00:00:00:00
 PeriodicAdvInt : 0x0000 (0)
 DataLength         : 0x0010 (16)
 Data               : 02:01:06:0C:09:50:72:6F:6A:65:63:74:5A:65:72:6F
Dump(Rx):
0000:04 FF 2F 13 06 00 00 00 40 00 01 00 BE 56 1F F2 ../.....@....V..
0010:F8 F0 01 02 00 7F C0 FF 00 00 00 00 00 00 00 00 ................
0020:10 00 02 01 06 0C 09 50 72 6F 6A 65 63 74 5A 65 .......ProjectZe
0030:72 6F                                           ro
------------------------------------------------------------------------
```

If you want, you can increase the size of the advertisement data.

The following image shows sniffer log capture of the advertisement event. In this case the advertisement data is 79 bytes. You can see the `ADV_EXT_IND` on channel 37, 38 and 39, followed by the `AUX_ADV_IND` on channel 24.



The next image is an overview of the advertisement mode. Here you can see the three horizontal rows of `ADV_EXT_IND` packets. These packets are always on channel 37, 38 and 39. On the other hand, the `AUX_ADV_IND` packets switch channel for each advertisement event. All secondary advertisement channels are used. This is the diagonal line of advertisement packets.



References

TI BLE5-Stack User's Guide (http://dev.ti.com/tirex/#/?link=Software%2FSimpleLink%20CC26X2%20SDK%2FDocuments%2FBLE5-Stack%2FBLE5-Stack%20User%27s%20Guide): PHY section located at **Developing a Bluetooth Low Energy Application → The Stack → Physical Layer (PHY)**

Bluetooth Core Specification (https://www.bluetooth.com/specifications/bluetooth-core-specification)

How does Bluetooth® 5 increase the achievable range of a Bluetooth low energy connection? (https://e2e.ti.com/blogs_/b/connecting_wirelessly/archive/2017/01/30/how-does-bluetooth-5-increase-the-achievable-range-of-a-bluetooth-low-energy-connection?tisearch=e2e-sitesearch&keymatch=%20user:49670)