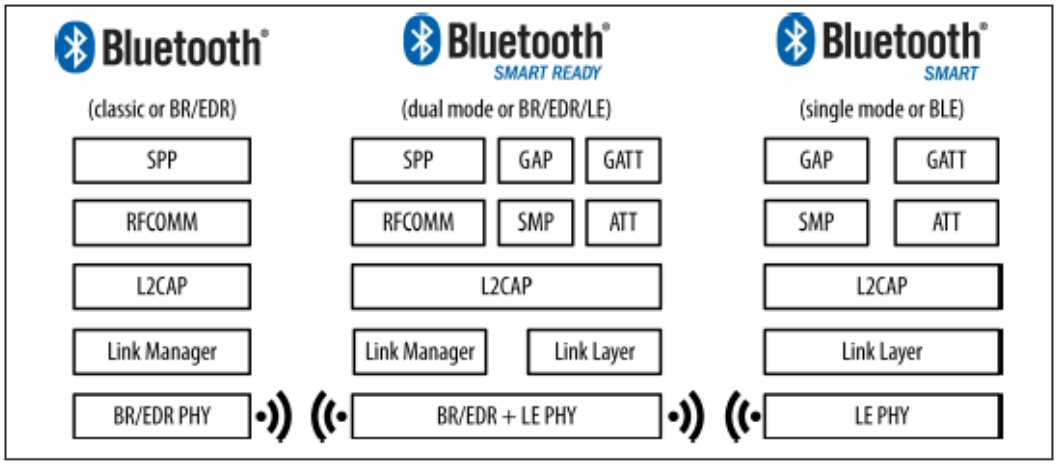
# Bluetooth for Serial Comm

 2015-07-01

## 1. Project Overview



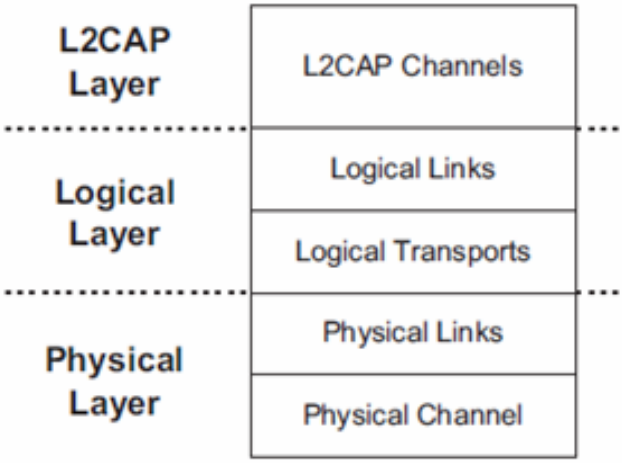

We have currently 2 types of Bluetooth
Classic Bluetooth with a range about 33 ft (10 m), theoretical max range is 330 ft (100m)
New LE (Low Energy) one (BLE) consumes less power to save hand-held battery, but with the same range, BLE has greater theoretical max range

The new BLE is NOT backward compatible with the classic one, so the Dual-Mode BT has both in the same module to support both. Both use the same spectrum range but different set of channels, the classic has 79 1-MHz channels, while BLE has 40 2- MHz ones.

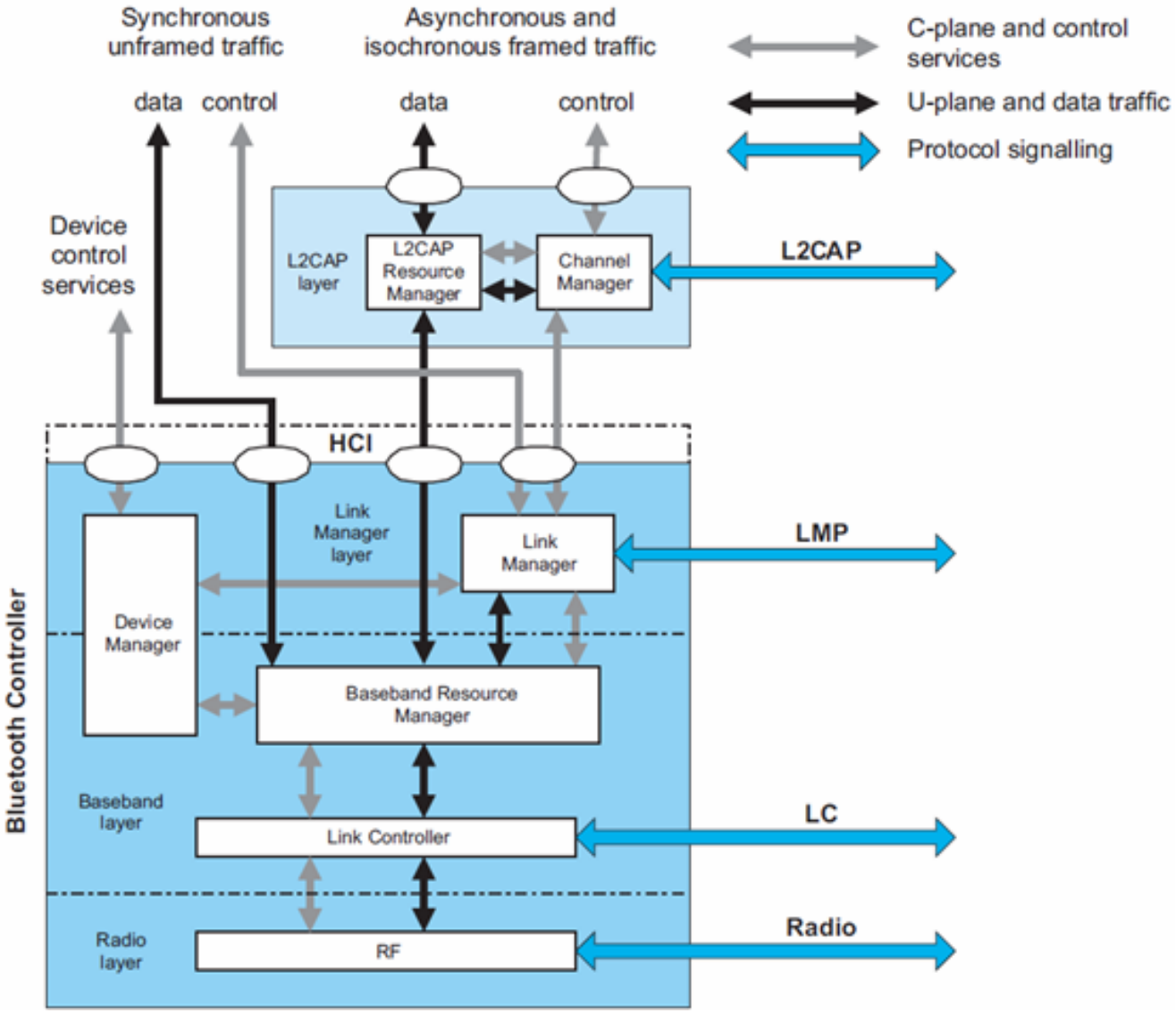
Project Scope
There are quite some BT modules in different formats, a CSR BT USB dongle is used in this project on target Microchip PIC32 Eva Kit as BT Server and on host PC (Linux & Win) as BT Client
Raw L2CAP packets are used via HCI ACL packet, NO use of BT profils. The standard BT serial link uses SPP profile via RFCOMM seen above
USB EP_0 is used for output BT commands, EP_1 for input BT Event, EP_2 for in/out L2CAP packets
Communication is based on BT MAC address (48-bit BD_ADDR), no discovery or pairing involved, in replace of classical PC serial comm. for user interface between target and host

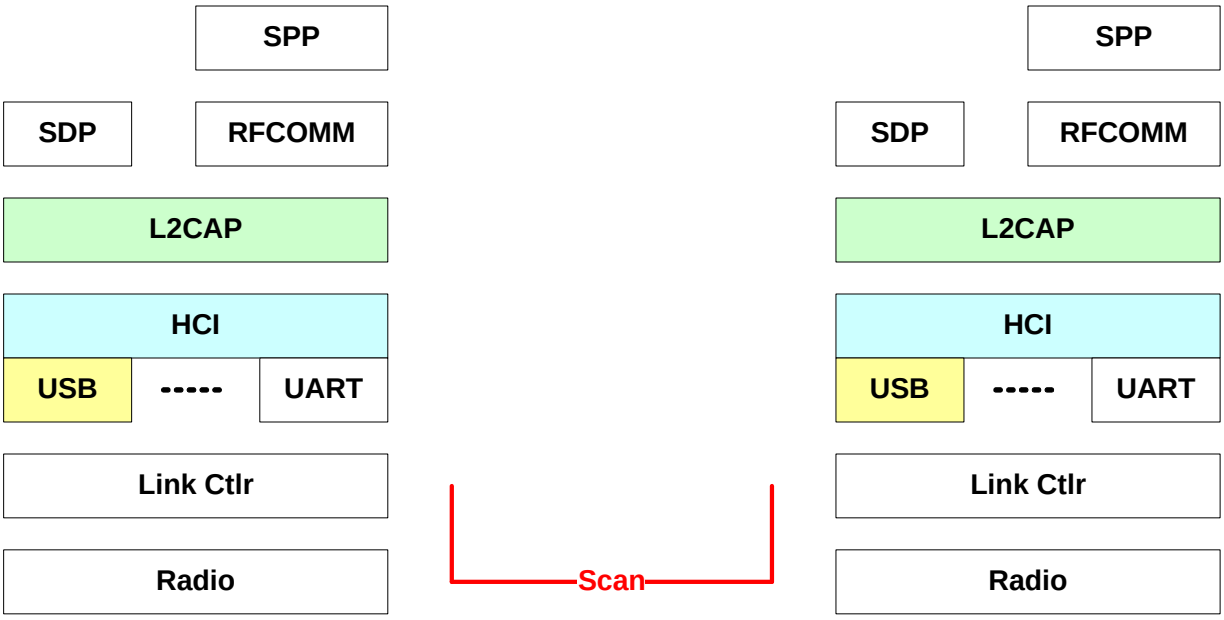
## 2. Bluetooth Overview

### 2.1. Bluetooth generic data transport architecture

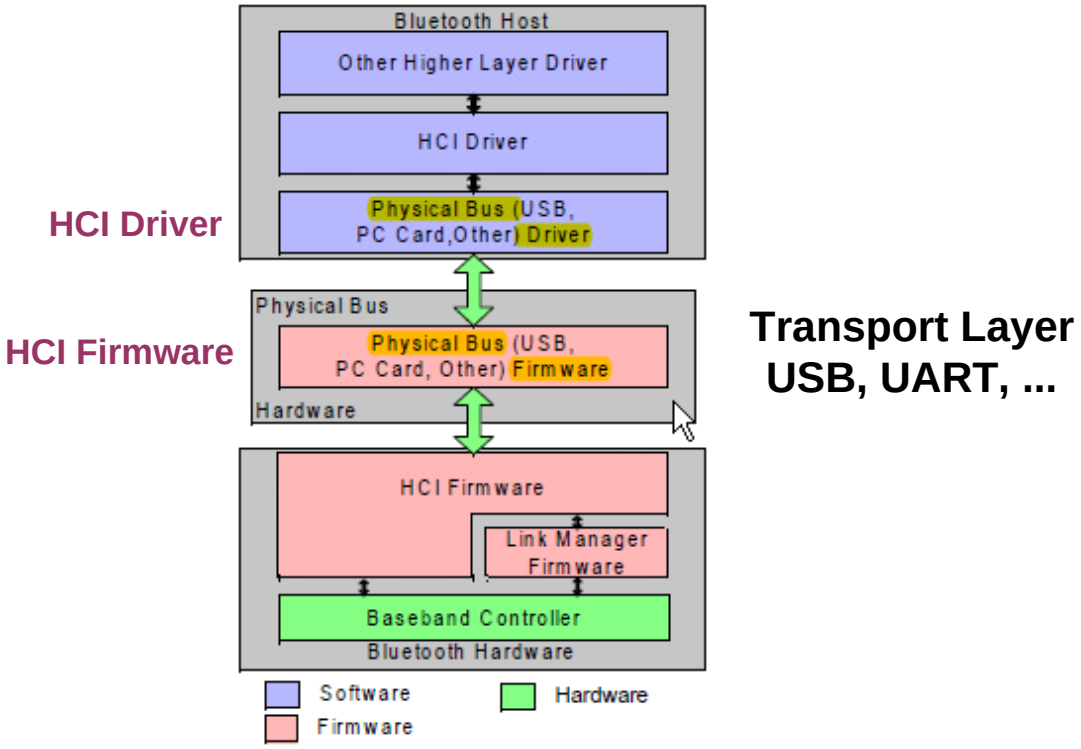



### 2.2. Bluetooth core system architecture



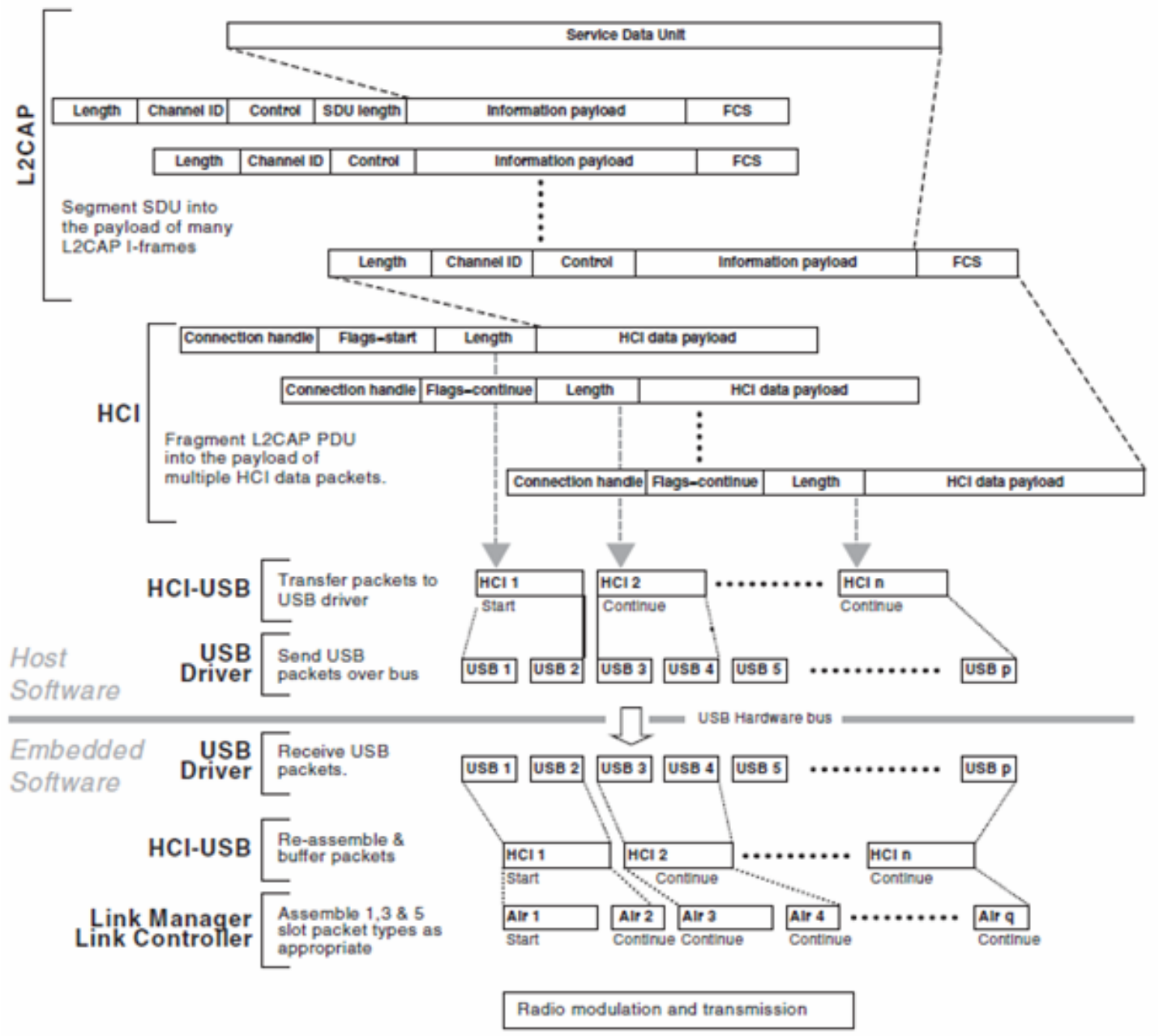



## 3. BlueTooth Stack




**Note**
Scan  stops at Link-Layer to get remote BDADDR, not higher layer, like USB, …

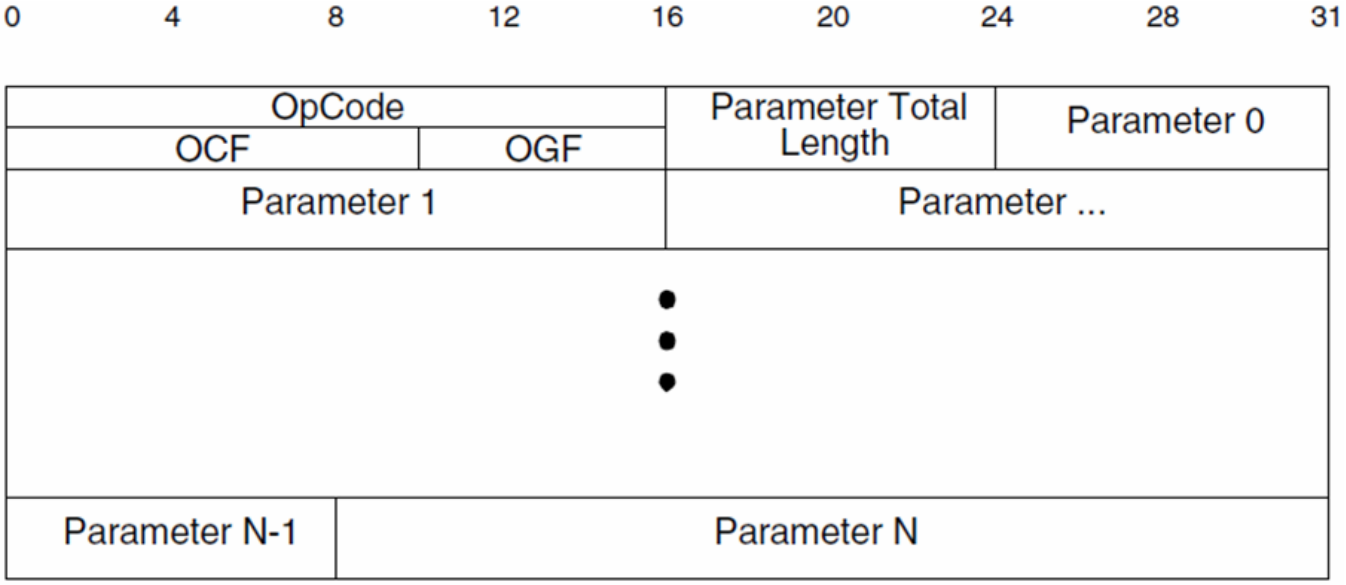
# 4. HCI Packet



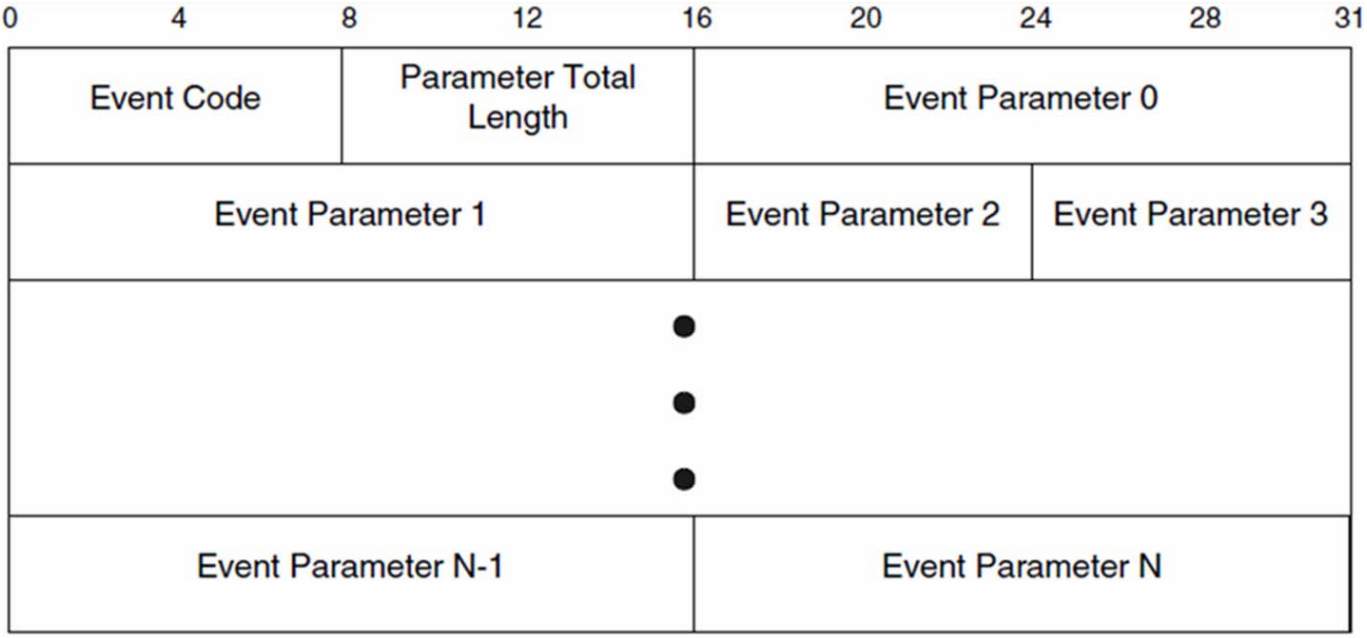

There’re 2 main types

- Command & Event packet : Opcode, Param_Len, Params, …
- Async & Sync Data packet : Connection Handle, Flags, Data Len, Payload

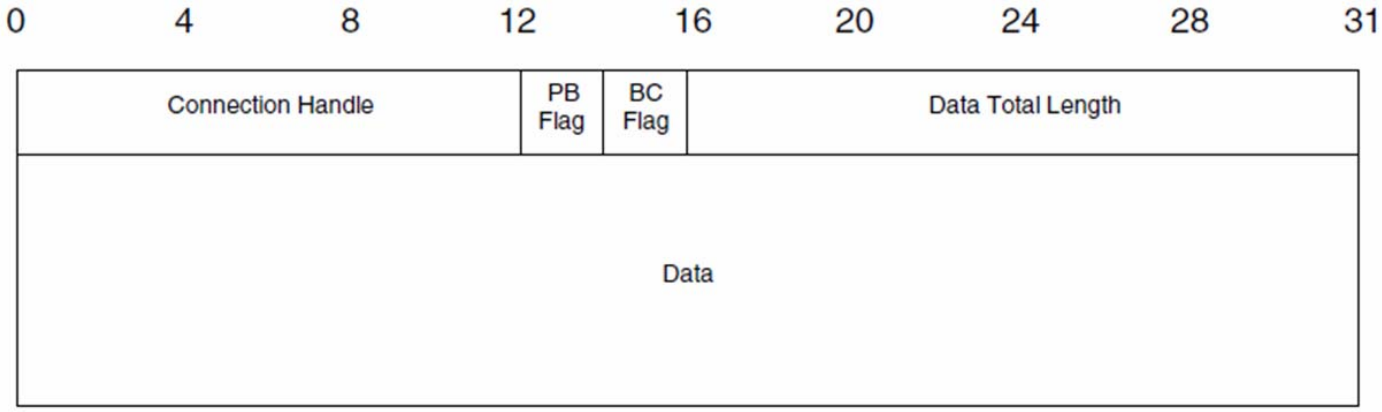
4.1. HCI Command Packet



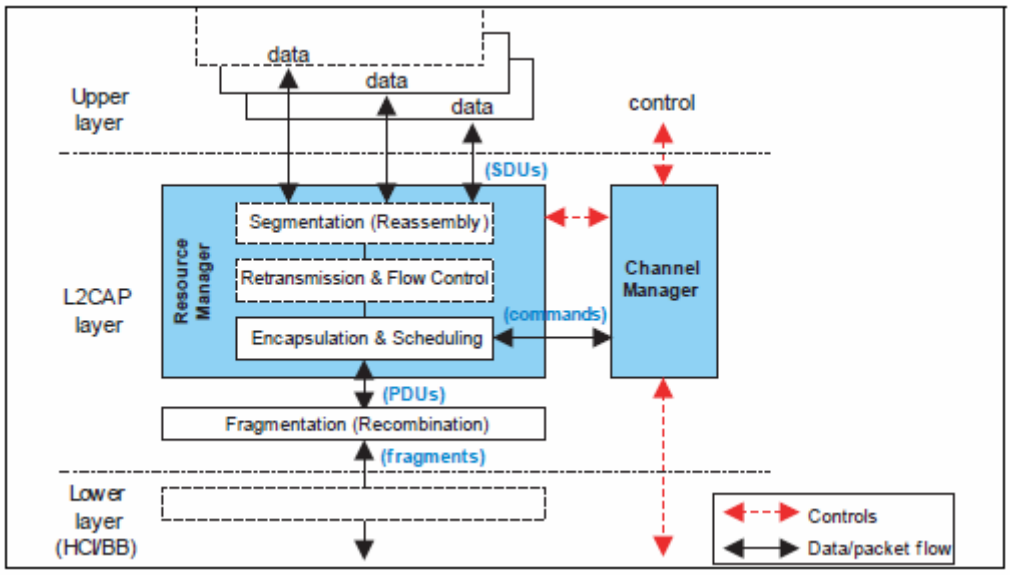

4.2. HCI Event

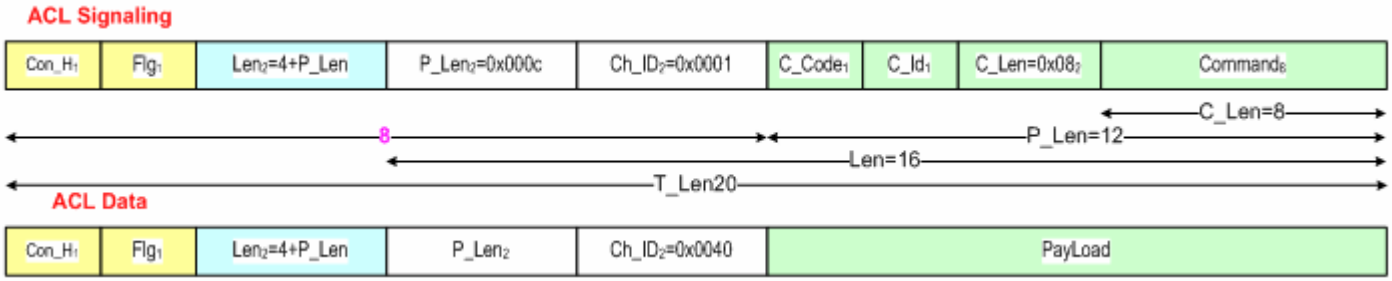



4.3. HCI ACL Data Packet (Async)



# 5. L2CP

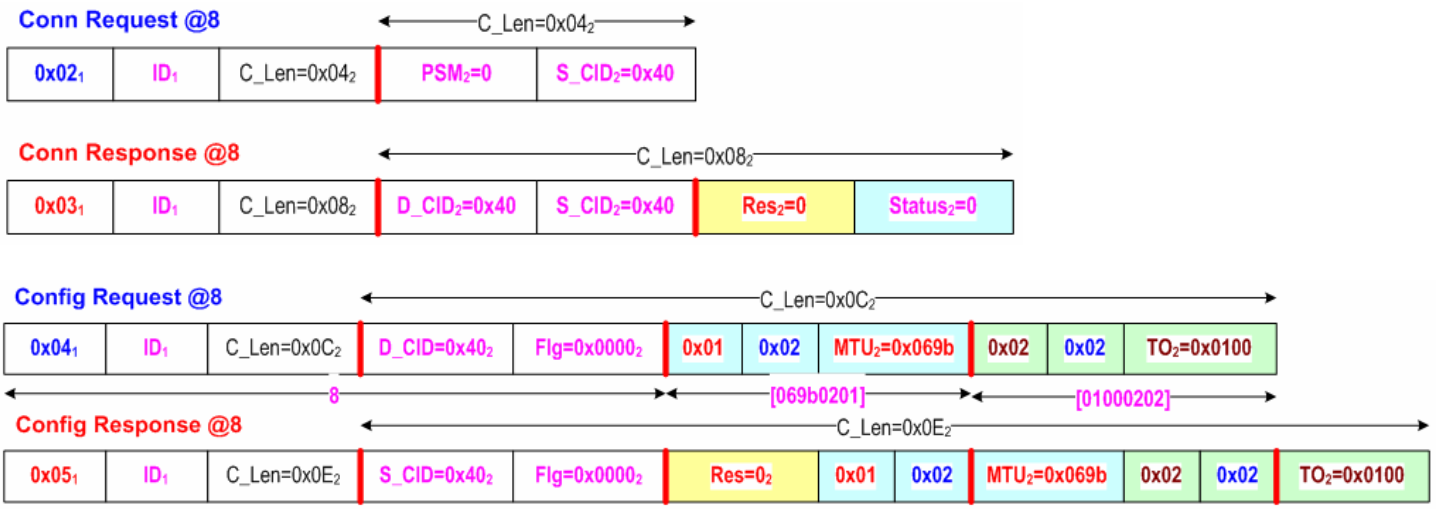



## 5.1. ACXL Signaling

| Cmd_Code | Field (2 bytes each) | Description |
|---|---|---|
| 0x00 | | RESERVED |
| 0x01 | Reason **(2)**<br>0<br>1<br>2<br>Data (opt) | Command Reject |
| 0x02 | PSM<br>1 : SDP<br>3 : RFCOMM<br>5 : Telephony Control Protocol<br>1001+ : Dynamic assigned<br>Src CID | Connection Request |
| 0x03 | Dst CID<br>Src CID<br>Result: 0-OK<br>Status: Pending only | Connection Response (8) |
| 0x04 | Dst CID<br>Flag | Configure Request (4) |
| 0x05 | Src CID<br>Flag<br>Result: 0-OK | Configure Response (8) |

| | Config | |
|---|---|---|
| 0x06 | Dst CID<br>Src CID | Disconnection Request (4) |
| 0x07 | Dst CID<br>Src CID | Disconnection Response (4) |
| 0x08 | Data (opt) | Echo Request |
| 0x09 | Data (opt) | Echo Response |
| 0x0A | InfoType<br>1 : Connectionless MTU | Information Request (2) |
| 0x0B | Info Type<br>Result: 0-OK<br>Data (opt) | Information Response (4+) |




## 5.2. Connection & Configuration Setup

A Bluetooth USB adapter is used for serial link between a host (Windows or Linux) as L2CAP Client and embedded target (BeagleBone, Microchip PIC32) as L2CAP Server.

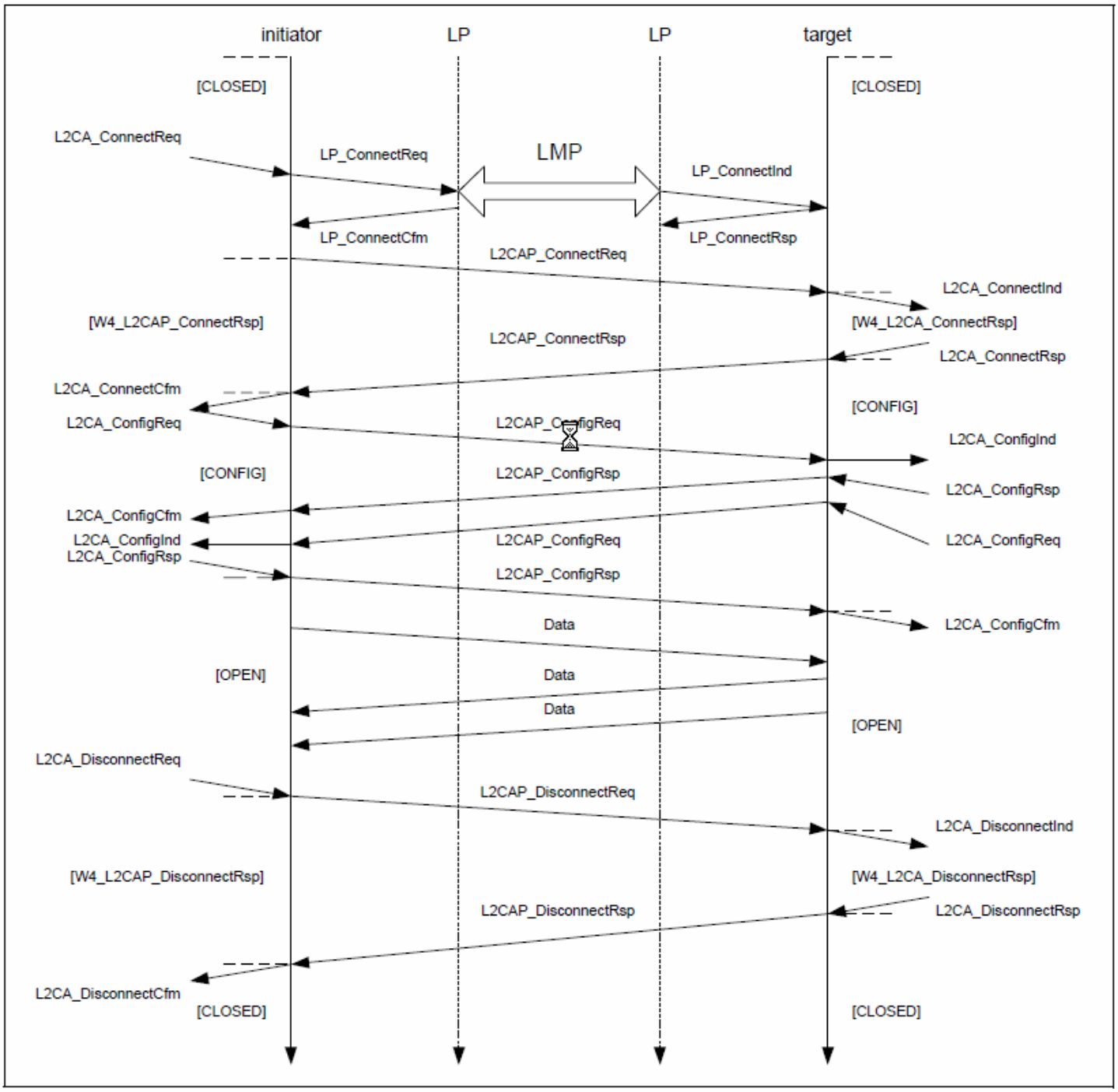
L2CAP is used for raw data exchange, there's NO use of any kind of Bluetooth profile.

In doing so, there're 2 phase to fulfil this task

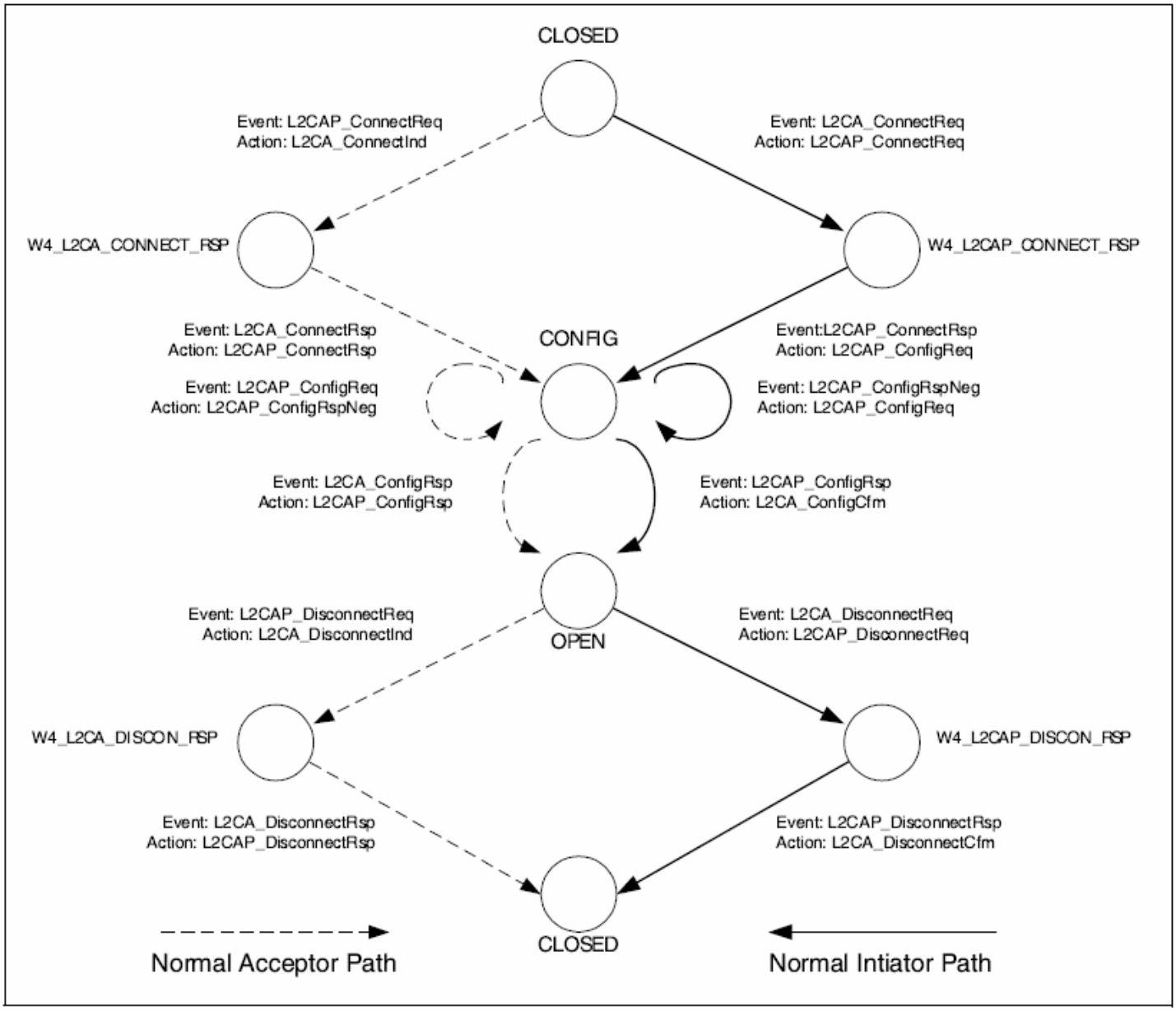
1) Basic BT HCI Connection Setup using BD_ADDR
    1. Reset BT module
    2. Read Buffer Size : it also gets number of packets allowed for data transfer
    3. Write Scannable

2) Basic L2CAP Connection & Configuration Setup
    1. Connection Request/Response
    2. Configuration Request/Response

## 5.3. Message Sequence Chart of Basic Operation



## 5.4. State Machine Example



# 6. Results

## 6.1. Test using L2PING on both Linux hosts

It's mysterious to me why L2PING stops after 10 pings with default 44-byte data packet!

Linusb & PinUsb-win32 are used for Client, but too slow for Server, and BT socket used instead.

## 6.2. Test using BeagleBone as Server and Linux Host as Client

A 300-byte packet is used in the test. It's less than BT buffer size 310 (HCI command Read_Buffer)

CSR BT is not reliable as Server on BB, so Broadcom BT is used instead.
Both CSR and Broadcom BTs are used on host as Client

| System Test | Packet Size | Number of Transactions before "reset by peer" |
|---|---|---|
| Linux/Linux (L2PING) | 300 bytes | 10 (Only TEN!) |
| **BT CSR**<br>Linux LibUsb<br>Windows LibUsb-Win32 | 300 | 64 |
| **BT CSR**<br>BeagleBone | | |
| **BT BroadCom**<br>Linux LibUsb (BroadCom)<br>Windows : Fail to recognize | 300 | 164 |
| **BT BroadCom**<br>BeagleBone | | |

## 6.3. Conclusion

The problem "reset by peer" could relate to ACL flushing time-out in BT USB transport layer as its value in Config Req appears has some impact on number of packet before the problem occurs