# LXSDF T2

LX Serial Data Format T2

Doc. ID. LXE12 V3

Release Date. 2018-02-07 .

*Abstract - Abstract - Simple packet and protocols easy to use. UART, Bluetooth, WiFi, Ethernet, etc. Extremely simple packet and easy to use. Multi channel real time stream transmission. This manual illustrates LXSDF T2 RX, TX data formats, code examples.*

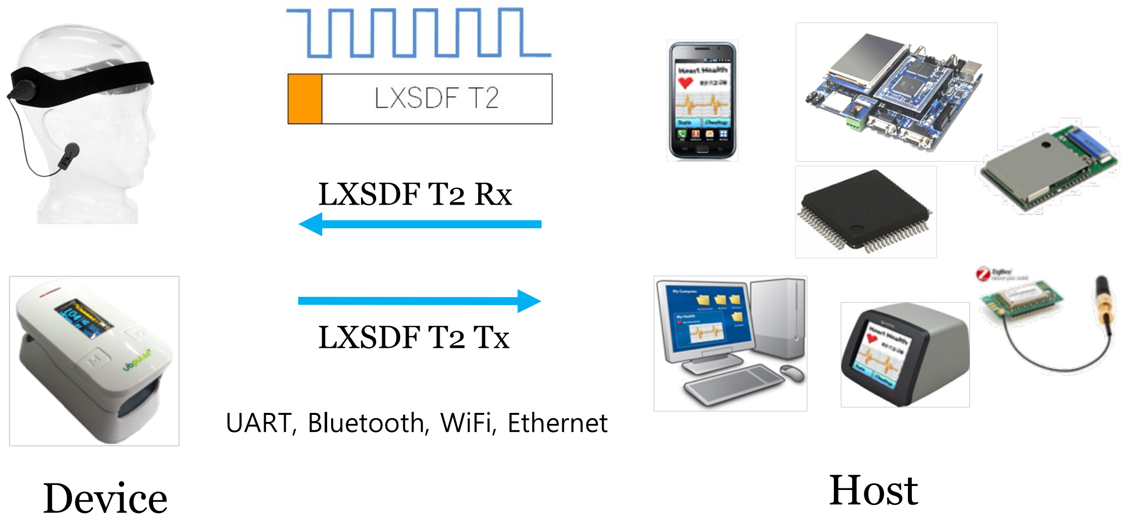
## Contents

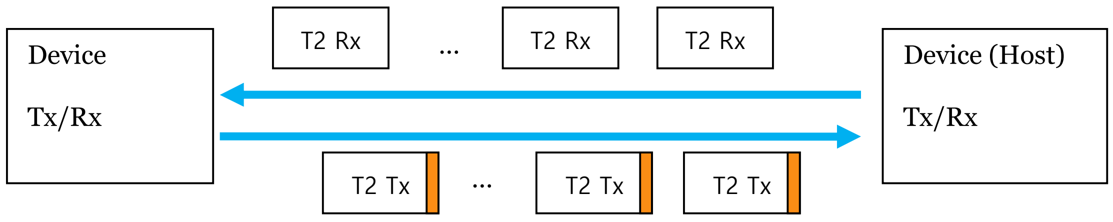
## LXSDF T2 Overview.

LXSDF T2 is *a g*eneral purpose serial communication standard which is able to transmit the real-time multi channel stream data and the low speed data in one packet. A typical example of multi channel stream is the time series data converted by the multi channel ADC(Analog to Digital Converter). T2 can be used in any serial communication system using UART (Universal Asynchronous Receiver & Transmitter), windows PC's com port, android host's serial port , wifi, bluetooth and ethernet etc.

UART(com port, serial port) is the most commonly used serial communication. UART can transmit one byte sequentially, but it rarely happens that sends only one byte of data element needed in Communication in real applied process. For example, when transmitting 12 bit AD converted data, must transmit serial communication 2 bytes dividing 4 bits and 8 bits and unify them into one 12bits data in receiver. If there are various kinds of data to transmit, it is in need of packet concept. LXSDF T2 can handle several bytes as one packet, transmitting and receiving sides use the data by separating according to the standard LXSDF T2.

As the following picture, the left side is device and the right side is host. In general, the devices have many cases that they transmit a lot of data to the host side, while the host side has the cases that they usually transmit simple orders. The data type transmitting from device to host (receiving from device as host) is named LXSDF T2 Tx and the data type receiving from host to device is named LXSDF T2 Rx.





LXSDF T2 Tx Packets and T2 Rx

## LXSDF T2 Packet

LXSDF T2 is defined differently between Tx and Rx. In this context, Tx and Rx mean transmission and reception respectively and the direction of communication as the device communicated with host. Tx means transmitting data from device to host. Rx means receiving the data from the host transmitted.

LXSDF T2 Rx data format is very simple because data contents transmitted from host to device is not asked to transmit large bytes while LXSDF T2 Tx data format transmitting data from device to hosts is able to transmit various information when it can handle series of dozens ~ hundreds bytes in one packet. The host side receiving the data from device needs the method to detect the starting spot of packet.

## LXSDF T2 Rx Data Format

| index | name | MSB Value |
|---|---|---|
| 0 | Cmd 0 | 1 |
| 1 | Cmd 1 | 0 |
| 2 | Cmd 2 | 0 |

**TABLE 1. LXSDF T2 Rx Format**

- LXSDF T2 Rx format uses only 3 bytes when transmitting data from host to device.
- The most significant bit (MSB) must be 1, 0, 0 per each bytes.
- Each index is one byte and transmitted sequentially in order when transmitting serial data.
- Those are defined differently in device as how to work by according to Cmd data.
- The LXSDF T2 Rx Cmd data must be defined on the product's communication specifications.

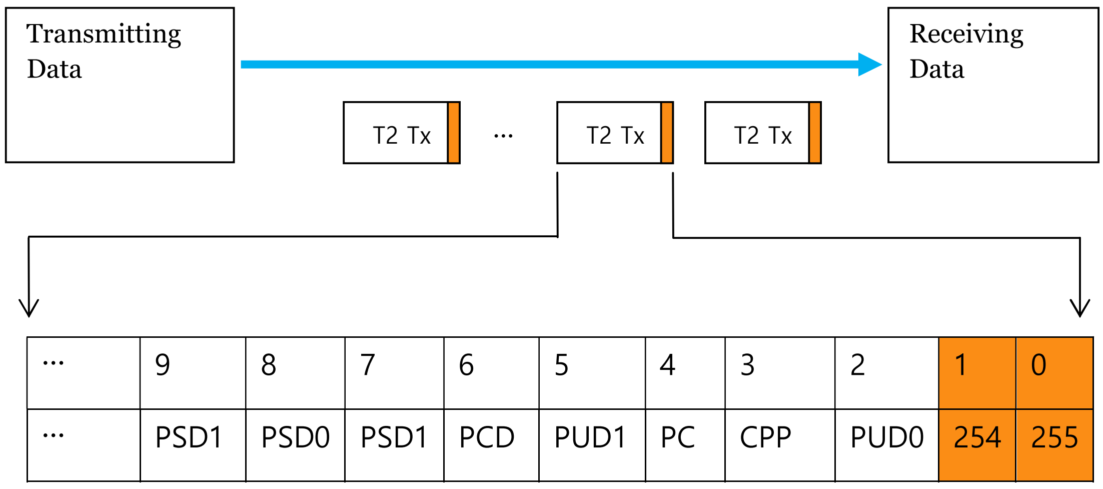
## LXSDF T2 TX Packet

### Sync Bytes : the key concepts of T2 Tx packet

LXSDF T2 Tx packet uses initial 2 bytes for each packet transmission as “Purpose for Synchronizing Packet”. The first byte is allocated the fixed value 255(0xFF in hex), and fixed 254 (0xFE in hex) for second byte. Namely, Sync Bytes are designed only in the spots where appear 255 and 254 sequentially in the whole packet byte arrays.

Receiving side should check each byte and extracts “Sync Bytes”, so it can find the starting spot of 1 packet. Once finding the starting spot, it is able to extract the data needed in the program under the TX packet Standard.

The following picture indicates that the orange color part in the first 1 packet is assigned “Sync Bytes” and then a series of bytes for one packet.



| … | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| … | PSD1 | PSD0 | PSD1 | PCD | PUD1 | PC | CPP | PUD0 | 254 | 255 |

## LXSDF T2 Tx Packet Definitions

The table shows the definitions of T2 Tx packet. Each index is one byte and transmitted in order when transmitting serial. An available value for each packet index, the terms of the packet elements.

| Index | Value | Packet Element Name |
|---|---|---|
| 0 | 255 | SyncByte0 (Synchronization Byte 0) |
| 1 | 254 | SyncByte1 (Synchronization Byte 1) |
| 2 | 0~255 | PUD0 (Packet Unit Data 0) |
| 3 | 0~127 | CRD (Command Response Data). bit 6<br>PUD2 (Packet Unit Data 2). bit 5,4,3<br>PCDT (Packet Cyclic Data Type). bit 2,1,0 |
| 4 | 0~255 | PC (Packet Count) |
| 5 | 0~127 | PUD 1 (Packet Unit Data 1) |
| 6 | 0~255 | PCD (Packet Cyclic Data) |
| 7 | 0~253 | PSD1 (Packet Stream Data High Byte) |
| 8 | 0~255 | PSD0 (Packet Stream Data Low Byte) |
| 9 | 0~253 | PSD1 (Packet Stream Data High Byte) |
| 10 | 0~255 | PSD0 (Packet Stream Data Low Byte) |
| … | … | … |
| N-1 | 0~253 | PSD1 (Packet Stream Data High Byte) |
| N | 0~255 | PSD0 (Packet Stream Data Low Byte) |

**Table 2. LXSDF T2 Tx Packet Definitions.**

| Color | Description |
|---|---|
| ▨ | Data placement for multi channel stream data. Freely expandable to any number of channels. The typical example of the multi channel stream data is the continuous output of a multi channel ADC(Analog to Digital Converter). |

## TX Packet Elements

### PC (Packet Count)

+1 for every one packet transmission and start 0 again after the maximum value.

By using PC, it is necessarily used to identify the data transmitted to PCD every packet.

The maximum of PC value gets different value according to PCD Type value. If PCDT is 0, PC maximum is 31.

### CRD (Command Response Data)

If the device receives the command from the other device, CRD value is reversed.

Usage – If CRD value is 1 before transmitting the order from host and the value is the same after transmitting, the order transmission is failed. If CRD value is changed, it means the device receives the order from host well.

### PUD 0, PUD 1, PUD 2 (Packet Unit Data)

Allocated data is different for each product. Mainly, information data to transmit at high speed is allocated.

### PCDT (Packet Cyclic Data Type)

PC maximum depends on PCDT value and data transmitted to packet cyclic data depends on PCDT value. PCDT is always 0 for the first stage(device power ON) and though PCD mode value is changed into different value like 1,2,3.. on the situation, it is changed into 0 automatically by completing data transmission of the mode one time.

| PCDT | PC (Packet Count) Maximum | Data |
|---|---|---|
| 0 | 31 | Exclusive data for LXSDF T2 and general data. |
| 1 | Depends on each products | |
| 2 | Depends on each products | |
| 3 | Depends on each products | |
| 4 | Depends on each products | |
| 5 | Depends on each products | |
| 6 | Depends on each products | |
| 7 | Depends on each products | |

## LXSDF T2A PCD Designated Data for PCDT 0

The section from PC 0 to 19 is for transmitting product's specialized data and the section from PC 20 to PC 31 is for system designated data. The system exclusive data is explained as below table.

| PCD[PC] | Item | Description |
|---|---|---|
| PCD[31] | Com port search information | fixed value 108. Information for searching device using LXSDF T2A. |
| PCD[30] | LXDeviceID | Allocated value between 1 and 255. Unique ID for identifying the device. |
| PCD[29] | ComFirmInfo1 | Firmware ID and version for processor 1. |
| PCD[28] | Number of channel | Number of channel from stream area of packet. |
| PCD[27] | Number of samples | Number of samples from stream area of packet. |
| PCD[26] | ComPath | Communication physical path. |
| PCD[25] | ComFirmInfo2 | Firmware ID and version for processor 2. |
| PCD[24] | ComFirmInfo3 | Firmware ID and version for processor 3. |
| PCD[23] | - | reserved |
| PCD[22] | - | reserved |
| PCD[21] | - | reserved |
| PCD[20] | - | reserved |

## ComPath

ComPath is used for mark to show what physical path to transmit data. It is possible to transmit more than two communication path in one device. The host received the data refers to Compath value to check the communication path.

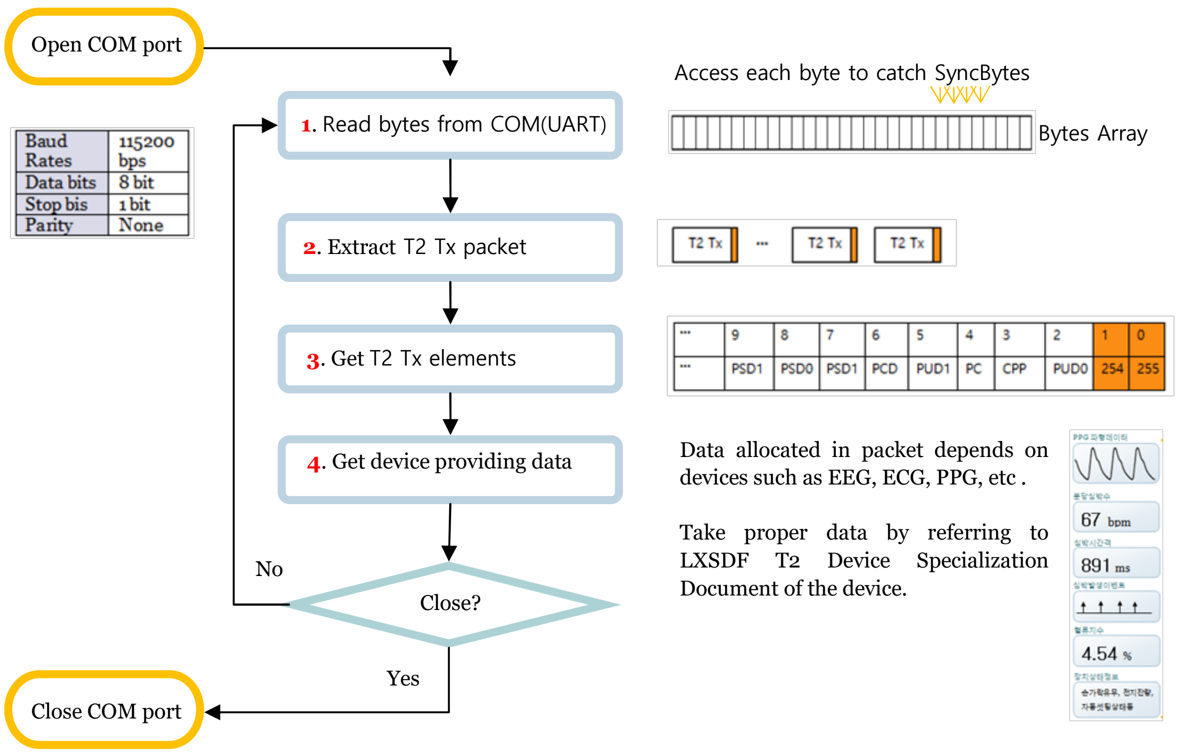
| Compath Value | Communication Path |
|---|---|
| 0 | UART |
| 1 | USB CDC |
| 2 | Bluetooth SPP(Serial Peripheral Profile) |
| 3 | Bluetooth Low Energy SPS |

## Host Programming guide

The following picture shows the programming flow of host side.

First of all, open com port in host. After that,

1. Read Bytes from COM : Read bytes received from com port(UART) in order.
2. Extract  T2 Tx packet : Catch Sync Bytes (it is placed in order of 255,254) which means packet's start spot from byte row separates data up and then abstract data elements in the packet from
3. Get T2 Tx element : Get all the T2A Packet elements.
4. Get device providing data : Data allocation situation depends on each device. Refer to LXSDF T2 Device Specialization Documents of the device.



| | |
|---|---|
| Baud Rates | 115200 bps |
| Data bits | 8 bit |
| Stop bis | 1 bit |
| Parity | None |

| ··· | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| ··· | PSD1 | PSD0 | PSD1 | PCD | PUD1 | PC | CPP | PUD0 | 254 | 255 |

Data allocated in packet depends on devices such as EEG, ECG, PPG, etc .

Take proper data by referring to LXSDF T2 Device Specialization Document of the device.

**Fig 1.  Program Structure in Host processing.**

## CODE EXAMPLE: LXSDF T2 Tx PACKET ABSTRACT AND PACKET DATA OCCUPANCY.

This code shows the example of  processing code of phase 2 and 3 in the picture. (C#code). This example just use simple general function so total coding method is the same regardless of the language.

```
// transmit the data received by serial with the function by 1 byte  in order .
// processing  in the function : Find the sync spot and abstract data by each packet TX Index .
bool Sync_After = false;
byte Packet_TX_Index = 0;
byte Data_Prev = 0; // PREVALUE

byte PUD0 = 0;
byte CRD_PUD2_PCDT = 0;
byte PUD1 = 0;
byte PacketCount = 0;
byte PacketCyclicData = 0;
byte psd_idx = 0;

int Parsing_LXSDFT2(byte data_crnt)
{
   int retv = 0;

   if (Data_Prev == 255 && data_crnt == 254)// Found sync spot.
   {
      Sync_After = true;
      Packet_TX_Index = 0;                 // Initialize  packet  TX Index  0.
   }

   Data_Prev = data_crnt;                  // receive the present value as pre value.

   if (Sync_After == true)                 // only task after discovering sync.
   {
      Packet_TX_Index++;                   // increase TX Index 1. The spot where is discovered 254 is 1. Whenever receiving 1 byte as serial, it
increases 1.
      if (Packet_TX_Index > 1)             // only task over TX Index  2.
      {
         if (Packet_TX_Index == 2)          // occupied TX Index2 PUD0.
            PUD0 = data_crnt;
         else if (Packet_TX_Index == 3)     // occupied TX Index 3 CRD, PUD2, PCD Type
        CRD_PUD2_PCDT = data_crnt;
         else if (Packet_TX_Index == 4)     // occupied TX Index 4  PC.
            PacketCount = data_crnt;
         else if (Packet_TX_Index == 5)     // occupied TX Index 5  PUD1.
            PUD1 = data_crnt;
         else if (Packet_TX_Index == 6)     // occupied TX Index 6 PCD(Packet cyclic data) .
            PacketCyclicData = data_crnt;
         else if (Packet_TX_Index > 6)      // Stream data(wave-pattern data) enters one each in order in more than TX Index 7. the data
procedure to receive ->  it is recorded in order of Ch 1 high byte , Ch 2 high byte low byte..
         {
            psd_idx = (byte)(Packet_TX_Index - 7); // Packet Stream Data arrangement Index.
            PacketStreamData[psd_idx] = data_crnt; // crnt data is occupied in order and stream data is only occupied.
            if (Packet_TX_Index == (Ch_Num * 2 * Sample_Num + 6)) // Channel number  x 2( 2bytes occupation) x Sample capacity + 6(Index
value before the front section of wave-pattern data) is the end of one packet.
            {
               Sync_After = false; // Be false to search sync spot again.
               retv = 1; // If Passing of 1 packet unit is finished, it will be returned.
            }
         }
      } //if (Packet_TX_Index > 1)

   }
   return retv; //If 1  packet is finished, others return 0.
}
```

**TABLE3. CODE EXAMPLE:  LXSDF T2 Tx PACKET ABSTRACT AND PACKET DATA OCCUPANCY**

# Appendix A. COM PORT AUTOMATIC SEARCH METHOD.

This explanations are for the device search in case device like PC. The PC attached device's com port number assignment is not fixed. If connecting MCU and UART in Embedded System, it doesn't need automatic search.

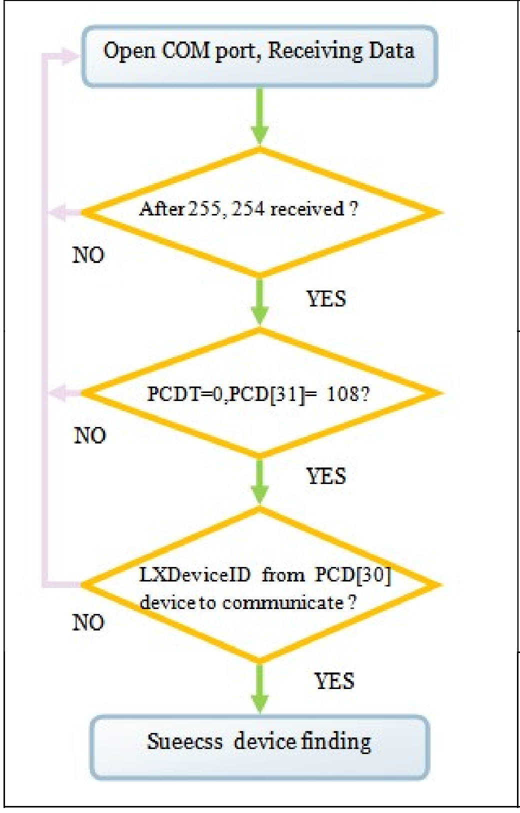
## Abstract

In case of device detecting COM port from host, there is case that user choose COM port to communicate in application program. This means bad products design regardless of user's convenience. It has to be designed that the program search COM port connected with device automatically. This function can not be solved only by software. It has to set the function to search automatically in the device.

## COM port search method.

In LXSDF T2 Tx packet, PCD "Com port search information" which is PCD Type 0 and PC=31 and PCD "LXDeviceID" which is PC=30 is used to find proper COM port of the product for information to use device search.

You can find the device's COM port easily if practicing the procures as the following explanation cycling all the Com port from host in order. Open one COM port temporarily and process the data received by byte unit like the following table.

| Flow Chart | Step | Description. |
|---|---|---|
|   Open COM port, Receiving Data<br>After 255, 254 received ?<br>NO<br>YES<br>PCDT=0,PCD[31]= 108?<br>NO<br>YES<br>LXDeviceID from PCD[30] device to communicate ?<br>NO<br>YES<br>Sueecss device finding | Step 1 | If 254 is detected next to 255 : It's possible device to communicate. Goto step 2.<br>If 254 is not detected next to 255 : it's not LXSDF T2 Tx packet. Start again opening another COM port. |
| | Step 2 | If PC (PACKET COUNT) value becomes number 31 -> It is sure of the device transmitting data to LXSDF T2 Format. However, it could transmit the same format data like LXSDF T2 in some products coincidentally. For occupying safely, if the packet cyclic data value in PC = 31 is 108, it is sure that is LXSDF T2 Tx packet. |
| | Step 3 | If device is communicated by LXSDF T2 Tx format, the next phase is to search model to communicate. At this time, check packet count 30 which is product's LXDeviceID to communicate. |

## COM PORT AUTOMATIC SEARCH C# CODE EXAMPLE.

Automatic search method to find the device is the same regardless of language whether it is C# or C++

```
        int bytestoread = sp.BytesToRead;   // occupied  byte number in Com port buffer. Sp is serial port object.

        // OUTPUT 1. Whether it is our device or Not? Our device must have the data in COM port..
        if (bytestoread == 0) { return; }  // If there is no data which can  be read in  COM port, this is not  LXSDF T2 format.
LXSDF T2 transmits the data every time.

        /// If there is some data to read in COM port, it reads all the data.
        byte[] rbuf = new byte[bytestoread]; // created the memory size dynamically.
        bool find_sync = false;
        sp.Read(rbuf, 0, bytestoread); //  received in rbuf tentatively..
        // OUTPUT 2. Check sync .
        for (int i = 0; i < bytestoread-1; i++) //
        {
           if (rbuf[i] == 255 && rbuf[i + 1] == 254) // Found the sync spot.
           {
              find_sync = true;
              break; // break the loop
           }
        }
        if (find_sync == false) return; // If there is no data in order of 255, 245, this is not  LXSDF T2.
        ///OUTPUT 3. Check the packet cyclic data in case of detecting some sync.  Must receive over certain time data
continuously to check it.
        byte[] cbuf = new byte[4096];
        int bytetoreadlimit =0;
        int readbytenum = 0;
        int sum_readbytenum = 0;
        bool while_continue = true;
        byte Packet_Count =0;
        byte PacketCyclicData = 0;
        bool find_108 = false;
        byte find_ComDeviceID = 0; // ComDeviceID allots more than value 1.
        byte find_NumChannel = 0;
        byte find_NumSample = 0;
        byte find_firmversion = 0;

        while (while_continue)
        {
           if(sp.BytesToRead > 4096)
              bytetoreadlimit = 4096;
           else
              bytetoreadlimit = sp.BytesToRead;

           readbytenum = sp.Read(cbuf, 0, bytetoreadlimit); // read the data and figure the byte cumulative sum.

           sum_readbytenum += readbytenum;

           for (int i = 0; i < readbytenum-3; i++)
           {
              if (cbuf[i] == 255 && cbuf[i + 1] == 254) // detected sync spot.
              {
                 Packet_Count = cbuf[i + 4];          // occupied packet count value.
                 PacketCyclicData = cbuf[i + 6];       // occupied packet cyclic data.
```

```
   if (Packet_Count == 31 &&  PacketCyclicData == 108)// If packet count is 31 and packet cyclic data is 108, it is surely
LXSDF T2 Type.
                        find_108 = true;
                     else if(Packet_Count == 30)                  // This spot  is for Product ID.
                        find_ComDeviceID = PacketCyclicData;
                     else if (Packet_Count == 29)                 // This spot is for firmware version number. It is necessary if
updating firmware by UART.
                        find_firmversion = PacketCyclicData;
                     else if (Packet_Count == 28)                 // Channel number transmitted into stream data.
                        find_NumChannel = PacketCyclicData;
                     else if (Packet_Count == 27)
                        find_NumSample = PacketCyclicData;

                     if (find_108 && find_NumSample > 0)  // This means loof break because  find_NumSample is in packet count
27 and  find_108 is in packet count 31. If both value were found , Medium value could be found.
                     {
                        while_continue = false;
                        break;
                     }

                  }
               }
               /// Designate the maximum value to review how many data can be received in COM port.  If this value is too big, it
takes very long time to search the device. So it's good to set the small value.
               /// To search the device by LXSDF T2 type , The minimum needed data capacity  must  be  at least 32 packets. In
other words, 68bytes ( byte capacity of 1 packet) x 32 = 2176 bytes.  It's possible to exam device search information because
it has 3000 bytes enough to be 32 packets.
               ///   Formula : byte capacity of 1 packet  can find the answer as 8 bytes + 64 bytes .
               /// 8 bytes : 1 packet is 8 bytes  from Tx Index 0  to 6
               /// 64  bytes :  Stream area is  channel number * 2(bytes) * sample number, though it has different value by each
product . Because the maximum channel number allotted from LXSDF T2 is 8 and sample number is within 4, the maximum
is  64 bytes.
               /// x 32 : must receive 32 packets to communicate packet count 0 to 31.
               if (sum_readbytenum > 3000)  // Forcing  Loof  break condition.
               {
                  while_continue = false;
                  break;
               }
            } // while
```

**TABLE 4. CODE EXAMPLE: COM PORT AUTOMETIC SEARCH.**

## Revision History

| Release Date | Doc. ID | Description of Change |
|---|---|---|
| 2018-02-07 | LXE12 V3 | more clear description. |
| 2017-02-14 | LXE12 V2 | url link modified. |
| 2012-12-03 | LXE12 V1.1 | 1. ADD TABLE LABLE. ADD TABLE LIST.<br>2. TABLE4. ADD ComPath data list and Reserved data list in PCD designated data of PCD Type 0 Page 12<br>3. ADD LXSDF T2 Rx and Tx CLASSIFICATION explanation and picture Page 4 |
| 2012-06-25 | LXE12 V1.0 | First release |