LXSDF T2 통신규격

LX Serial Data Format Type 2 스트림 및 일반데이터 동시 전송 범용 시리얼통신규격

Doc. ID. LXD12 V2

Release Date. 2017-02-14 .

Abstract - LXSDF T2(LX Serial Data Format Type2) 는 실시간 스트림데이터 전송과 동시에 상대적으로 저속인 일반 데이터들을 하나의 패킷형식으로 전송 가능한 범용의 시리얼 통신포맷이다. 스트림데이터란 아날로그 신호의 AD 변환된 시계열 데이터 류가 대표적인 예이다. 본 글에서는 LXSDF T2 의 RX 데이터 포맷, TX 패킷구조를 설명한다. 또한 호스트측에서 TX 패킷을 수신처리 하는 코드 예, 장치와 연결된 COM 포트를 자동으로 탐색하는 법에 대한 설명과 코드 예를 제시한다.

목차

그림 목차.

표 목차.

LXSDF T2 개요.

LXSDF T2(LX Serial Data Format Type2) 는 스트림데이터의 실시간 전송과 동시에 상대적으로 저속인 기타 데이터들이 1 개의 패킷형식내에서 전송 가능한 범용의 시리얼 통신포맷이다. 스트림데이터란 아날로그 신호의 AD 변환된 시계열 데이터와 같이 시간적으로 상관된 데이터를 의미한다. 시리얼 통신 이라함은 임베디드분야에서는 UART(Universal Asynchronous Receiver & Transmitter), PC 나 스마트 폰 등에서는 COM 포트, 시리얼포트, Rs232 등으로 불리는 것을 지칭한다.

시리얼 통신 규격은 보통 1 바이트를 기본 전송단위로 하여 반복적으로 전송되는 형식이나, 실제 응용과정에서는 통신에 요구되는 데이터 요소가 1 바이트 단위로만 반복 전송하는 경우는 극히 드물다. 예로 12 비트 AD 변환 샘플링 데이터를 전송해야 하는 경우에는 4 비트, 8 비트로 분리해서 시리얼 통신 2 개 바이트에 분리하여 전송해야하며, 수신한 측에서는 이를 다시 1 개의 데이터로 통합해야한다. 만일 전송해야할 데이터 종류가 다양한 경우에는 패킷개념이 요구된다. 여러 개의 바이트를 1 개의 패킷으로 핸들링 할 수 있는 규격이 요구되며, 이 규격에 따라 송신하고, 수신한 측에서는 규격에 따라 수신한 데이터를 분리해서 활용하게된다. 패킷형식의 규격은 사용하는 용도에 따라 다양한 규격이 있을 수 있으며, LXSDF T2 에서는 고속의 실시간 다채널 실시간 파형 데이터를 전송 가능함과 동시에 장치의 상태정보, 계산값등과 같은 저속의 데이터도 동일한 1 개의 패킷 형식으로 모두 전송 가능한 범용의 실시간 통신 활용에 적합한 시리얼 통신 데이터 포맷이다.

LXSDF T2 의 Rx, Tx 구분

LXSDF T2 는 Rx, Tx 2 종류의 데이터 포맷이 따로 정의 되어 있다. 여기서 Rx, Tx 는 각각 수신, 송신을 의미하며 호스트와 통신하는 장치 입장에서의 통신의 방향성을 의미한다. 아래 그림에서 왼쪽이 장치의 예이고 오른쪽이 호스트의 예이다. 호스트로는 일반 PC, 스마트폰, 일반 임베디드 시스템등이 있다. 보통 장치들은 호스트측으로 많은 데이터를 전송해야 하는 경우가 많고 반면 호스트 측에서 장치 쪽으로는 간단한 명령이 전송되는 경우가 대부분이다. 장치에서 호스트로 송신(호스트 입장에서는 장치로부터 수신)하는 데이터 형식을 LXSDF T2 Tx 패킷이라고 지칭하고, 장치가 호스트로부터 수신받는 데이터 형식을 LXSDF T2 Rx 라고 지칭한다.

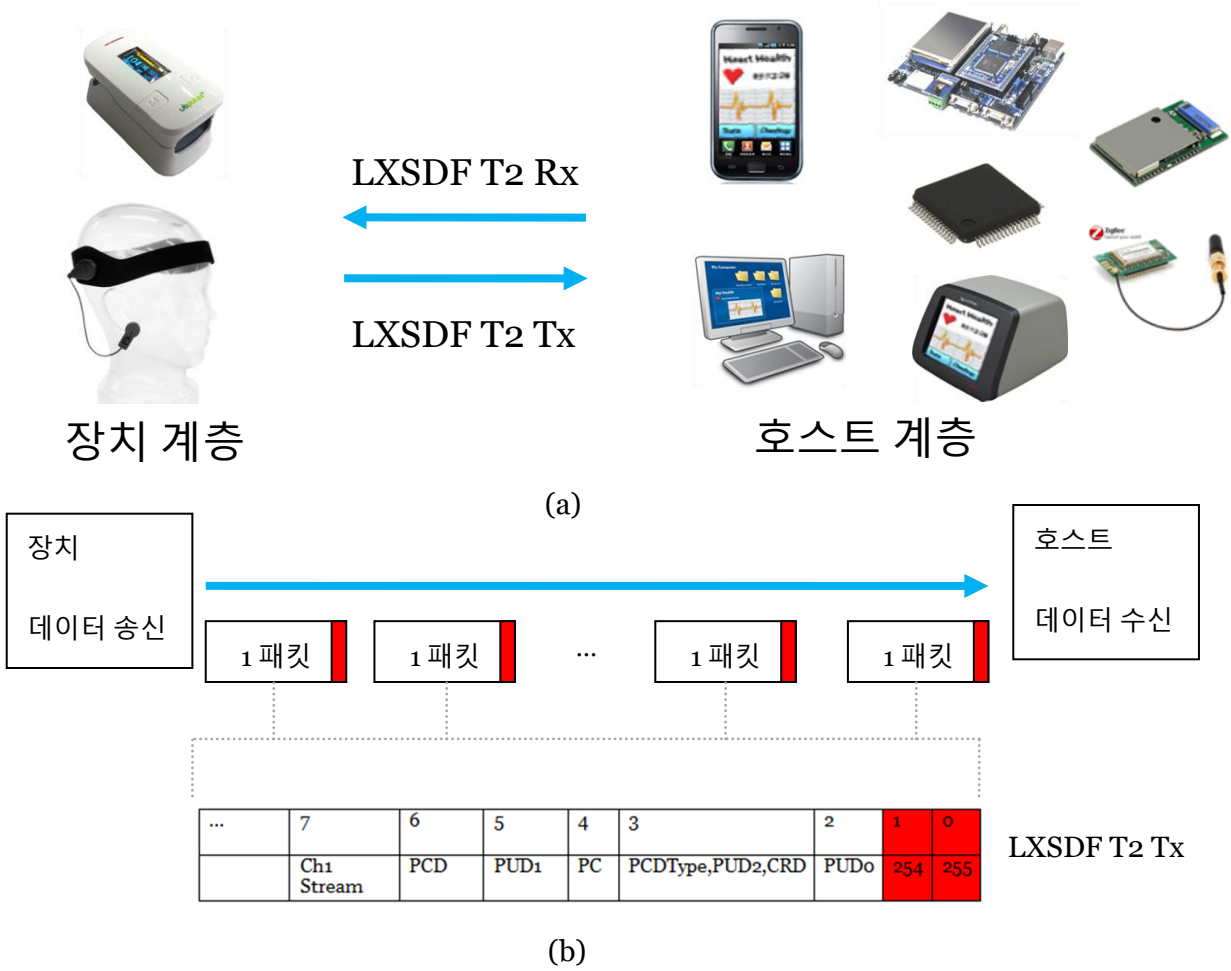


그림 1. LXSDF T2 의 Rx, Tx 구분, Tx 패킷의 세부 구조.

LXSDF T2 Rx 데이터 포맷.

호스트에서 장치로 데이터를 전송할 때 사용되는 데이터 규격이다.

연속한 3 바이트를 사용하며, Rx 인덱스 0,1,2 순서로 호스트에서 데이터를 전송해야 한다. 시간적으로 먼저 전송되는 Rx 인덱스 0 의 최상위 비트는 1, 나머지 2 개 바이트의 최상위 비트에는 반드시 0 이 기록되어 있어야 한다. Cmd 데이터에 따라서 장치에 어떤 작용을 하는지는 제품별로 달리 정의되며 해당제품의 통신 규격을 참조 해야 한다.

Rx 인덱스	명칭	규칙.
0	Cmd 0	최상위 비트 = 1
1	Cmd 1	최상위 비트 = 0
2	Cmd 2	최상위 비트 = 0

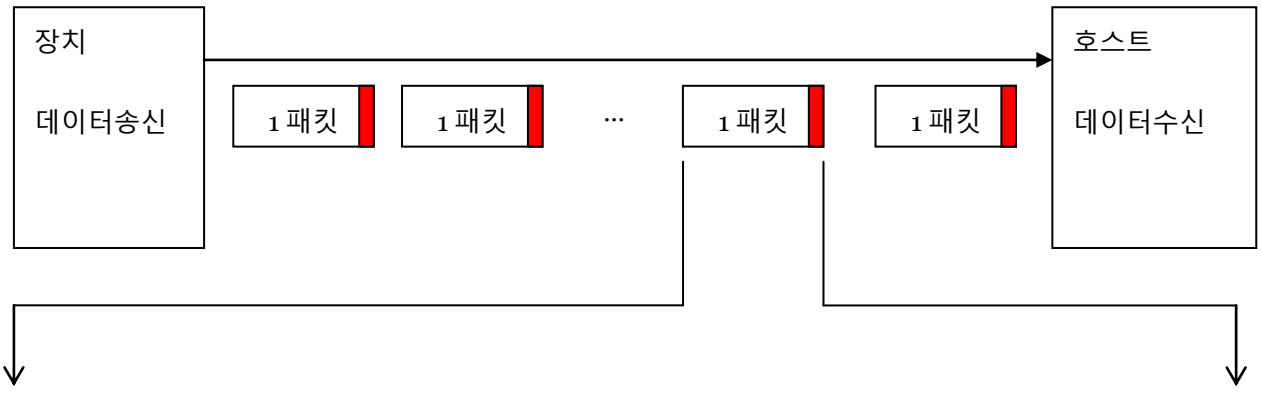
표 1. LXSDF T2 Rx Format

LXSDF T2 의 TX 패킷

LXSDF T2 는 장치에서 스마트폰 등의 호스트로 데이터를 전송하는 (장치 입장에서 송신용 즉 TX) 부분과 호스트에서 장치로 데이터를 전송하는 (장치 입장에서 수신용 즉 RX) 부분의 데이터 규격이 각각 다르게 정의된다.

호스트에서 장치로 전송할 데이터 내용은 길지 않은 바이트 전송이 요구되므로 RX 데이터 포맷은 매우 간단한 반면, 장치에서 호스트로 전송하는 TX 데이터 포맷은 일련의 수십~수백개의 바이트가 1 개의 패킷으로 취급할 수 있어야 다양한 정보 전송이 가능하다. 장치에서 송신한 데이터를 수신하는 호스트측에서는 패킷의 첫시작점을 파악할 수 있는 수단이 필요하다. LXSDF T2 TX 패킷에서는 동기바이트로 각 패킷 전송 초기 2 개 바이트를 사용하며, 첫 1 바이트에는 255, 연속해서 다음 바이트에는 254 의 값이 기록되어있다. 즉 패킷 전체 바이트열 중에서 연속해서 255 와 254 가 등장하는 지점은 동기바이트가 유일하도록 설계되어 있다. 장치에서 데이터 전송시 항상 이 규격에 맞게 데이터를 호스트로 전송한다. 수신측에서는 전송되어 오는 각각의 단위 바이트들을 상시 모니터링 하여 동기 바이트를 검출하는 것으로 1 패킷의 시작점을 찾을 수 있다. 시작점을 찾게 되면 LXSDF T2 TX 패킷 규칙에 의거하여 필요한 데이터들을 프로그램에서 추출할 수 있게 된다.

아래 그림에서 1 패킷의 가장 첫부분에 빨간색으로 표시된 부분에는 항상 패킷의 첫지점을 나타내는 동기바이트가 2 바이트 할당 되어 있고, 그 이후에 일련의 바이트 단위의 데이터들이 연속 전송된다.



…	7	6	5	4	3	2	1	0
	Ch1 Stream	PCD	PUD1	PC	PCDType,PUD2,CRD	PUD0	254	255

LXSDF T2 Tx

LXSDF T2 TX 패킷 형식.

각각의 인덱스가 1 바이트를 점하며, 시리얼 전송시 0,1,2,... 의 순서로 전송하게 된다.

TX 인덱스	패킷요소	설명	값
0	Syncbyte 0	동기바이트 0	**255**
1	Syncbyte 1	동기바이트 1	**254**
2	**PUD 0** (Packet Unit Data 1, 패킷단위데이터 1)	Bit7 ~Bit0 : 일반데이터.	0~255
3	**CRD** (Command Response Data 명령응답데이터)	Bit6. 호스트측에서 보내온 RX 데이터를 장치에서 수신한 경우 비트 상태 반전.	Bit7=0
	PUD 2 (Packet Unit Data 2 패킷단위데이터 2)	Bit5,4,3 : 일반데이터.	
	PCD Type (Packet Cyclic Data Type, 패킷순환데이터 모드)	Bit2,1,0 : 이 값이 다르면 패킷순환데이터의 타입이 다름을 의미한다.	
4	**PC** (Packet Count)	매번의 패킷 전송시 1 씩 증가. 최대값 이후 다시 0 부터 시작.	0~255. PCD Type 에 따라 최대값은 변경됨.
5	**PUD 1** (Packet Unit Data 0 패킷단위데이터 0)	Bit6~Bit0 : 일반데이터	Bit7 = 0
6	**PCD** (Packet Cyclic Data, 패킷순환데이터)	PCD 에는 PCD Type 으로 PCD 의 타입이 지정되며, Packet Count 를 PCD 식별용으로 사용한다. **주 1**	0~255
7	Ch1 Stream high	Bit6,5,4 : 일반데이터. 하위 4 비트: 스트림 채널 1 의 상위 4 비트	Bit7= 0

8	Ch1 Stream low	스트림 채널 1 의 하위 8 비트	0~255
9	Ch2 Stream high	Bit6,5,4 : 일반데이터. 하위 4 비트: 스트림 채널 2 의 상위 4 비트	Bit7 = 0
10	Ch2 Stream low	스트림 채널 2 의 하위 8 비트	0~255
11	Ch3 Stream high	Bit6,5,4 : 일반데이터. 하위 4 비트: 스트림 채널 3 의 상위 4 비트	Bit7 = 0
12	Ch3 Stream low	스트림 채널 3 의 하위 8 비트	0~255
13	Ch4 Stream high	Bit6,5,4 : 일반데이터. 스트림 채널 4 의 상위 4 비트	Bit7 = 0
14	Ch4 Stream low	스트림 채널 4 의 하위 8 비트	0~255
15	Ch1 Stream high	Bit6,5,4 :일반데이터. 스트림 채널 1 의 상위 4 비트	Bit7 = 0
16	Ch1 Stream low	스트림 채널 1 의 하위 8 비트	0~255
..			
70 (주 2)	Ch4 Stream low		0~255

표 2. LXSDF T2 Tx Format

주 1

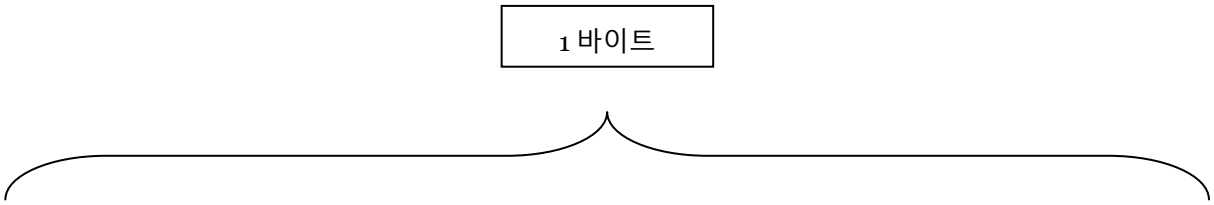
패킷단위데이터가 패킷카운트별로 데이터의 종류는 LXSDF T2 포맷이 응용되는 제품마다 별도로 정의되며, PCD Type 에 따라서도 다른 종류의 데이터 전송 가능.

주 2

위 표의 스트림데이터는 채널 4 개, 1 패킷당 전송되는 샘플수량은 1 인 경우를 예로서 나타낸 것이며, LXSDF T2 는 채널수, 샘플수량이 유동적인것도 전송가능하다.

용어 - 숫자비트, 비트숫자.

비트의 수량을 표현할 때는 “숫자비트”로 표현하고, 반면에 비트 요소를 지칭할때는 “비트숫자”로 표현한다. 예로, “7 비트”라 함은 “7 개의 비트”를 의미하고 “비트 7” 이라 함은 아래 그림에서의 최상위 비트를 의미한다. “비트 0”이라 함은 최하위 비트를 의미한다.



MSB(최상위비트) LSB(최하위비트)

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

LXSDF T2 TX 패킷 전송 주기.

장치 전원이 켜지면 장치는 LXSDF T2 Tx 형식의 데이터를 연속으로 전송한다.

LXSDF T2 Tx 패킷을 장치에서 전송할 때 패킷 전송주기는 제품마다 다르게 설정될 수 있으며, 전송되는 주기는 보통 초당 128 번, 256 번, 512 번, 1024 번으로 패킷을 전송한다.

TX 패킷요소별 설명.

PC(Packet Count)

매번의 패킷 전송시 마다 1 씩 증가되는 최대값 이후 다시 0 부터 시작.

PC 를 이용하여 매번의 패킷마다 PCD 로 전송되는 데이터가 어떤 데이터인지 식별함에 필수로 사용된다.

PC 의 최대값은 PCD Type 값에 따라 다른 값을 갖게 되며 PCD Type = 0 인 경우 PC 의 최대값은 31 이다.

CRD(Command Response Data)

호스트 측에서 LXSDF T2 Rx 형식으로 명령을 장치로 전달한 것을 장치에서 수신 성공한 경우 CRD 값을 반전한다.

활용 – 호스트에서 명령을 전송하기 전의 CRD 값이 1 이었는데 명령 전송후에도 1 이라면 명령전달 실패, CRD 값이 반전되어 0 으로 변경되었다면 호스트에서 전송한 명령을 장치에서 수신 성공했음을 의미한다.

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

제품마다 할당되는 데이터가 다르다. 주로 고속으로 전송해야할 정보류의 데이터가 할당된다.

PCD Type (Packet Cyclic Data Type)

이 값에 따라 패킷카운트의 최대값이 달라지며, 또한 PCD Type 값에 따라 패킷순환데이터로 전달되는 데이터가 달라진다.

장치가 켜진 초기상태는 PCD Type 값은 항상 0 이며, 상황에 따라 PCD 모드 값이 1,2,3 등의 다른 값으로 변경되어도, 해당모드의 데이터 전송이 1 회 완료 되면 다시 자동으로 0 으로 변경된다.

데이터 전송의 1 회 완료란 PC = 0 에서 시작하여 PC 의 최대값이 될 때까지가 1 회 완료다.

PCD Type	PC 최대값 및 총소요시간	데이터
0 (000)	31, (주 1)	LXSDF T2 포맷 전용 데이터 및 일반데이터 (전용 데이터는 아래 별도 설명)
1 (001)	제품마다 다름	
2 (010)	제품마다 다름	
3 (011)	제품마다 다름	
4 (100)	제품마다 다름	
5 (101)	제품마다 다름	
6 (110)	제품마다 다름	
7 (111)	제품마다 다름	

표 3. PCD Type 별 데이터배치.

(주 1) 총 소요시간.

PC = 0 인 패킷이 전송되고 난 이후 PC 값이 증가되어 PC 최대값을 전송한 후 다시 처음의 0 번 패킷을 전송하는데 소요되는 총 시간. (제품마다 패킷전송주기는 다름)

총 소요시간 계산식.

초당 N 번 패킷을 전송하고, PC 최대값이 M 인 경우 : 총 소요시간 = (1/N) x (M+1) 초

예 : 초당 250 번인 패킷을 전송하고 PC 최대값이 31 인 경우 (1/250) x 32 = 0.128 초.

PCD Type 0 의 PCD 시스템 지정 데이터.

PCD Type 0 의 PC 0 에서 23 까지는 제품 특화된 데이터를 전송할 수 있는 구간이며, PC 24 에서 PC 31 까지는 시스템 지정 데이터 영역이다.

항목	설명	비고	PC(패킷카운트).
COM 포트탐색정보	호스트에서 장치 탐색시 활용되는 정보.	108 이 고정기록됨.	31
ComDeviceID (제품고유번호)	제품별로 식별가능한 고유번호. LXSDF T2 포맷을 이용하는 제품은 여러 개가 있을 수 있으며, 해당제품의 고유번호가 전송된다.	1~255 사이의 값이 할당됨.	30
펌웨어정보	제조사 관리 펌웨어정보	1 바이트.	29
채널수	패킷의 스트림영역에서 전송되는 채널수	1 바이트.	28
샘플수	패킷의 스트림영역에서 전송되는 샘플링 수.	1 바이트.	27
기타정보 1	하위 3 비트 : ComPath	1 바이트.	26
Reserved	제품 특화 데이터 배치 금지.	1 바이트	25
Reserved	제품 특화 데이터 배치 금지.	1 바이트	24

표 4. PCD Type 0 의 PCD 지정 데이터.

기타정보 1 의 ComPath 는 PC26 의 PCD 1 바이트 중 하위 3 비트에 할당되며, 데이터가 어떤 통신경로로 전송되었는지 표식용으로 사용된다. 이는 1 개의 장치에서 2 개 이상의 통신경로로 LXSDF T2 형식의 데이터 전송이 가능하며, 이를 수신한 호스트 측에서 통신경로를 확인하고자 할 때 ComPath 의 값을 참조한다.

ComPath 값	의미	설명.
0	유선 UART	장치에서 UART 로 송신한 데이터 패킷이다.
1	유선 USB CDC	장치에서 USB CDC 로 송신한 데이터 패킷이다.
2	무선 Bluetooth SPP	장치에서 블루투스 SPP 로 송신한 데이터 패킷이다.

장치와 통신하는 호스트측의 프로그램 구조.

LXSDF T2 형식의 통신포맷을 사용하는 장치와 통신하기 위한 호스트측의 프로그램 전체 구조를 그림에 보이고 있다. 호스트에서는 가장 먼저 COM 포트 오픈 하는 것으로 시작한다. 임베디드 시스템에서는 MCU 에서 UART 통신설정을 해둬야 한다. 이후 "1. com 포트 (UART)에서 데이터 읽기"에서는 순차적으로 com 포트(UART)에서 수신된 바이트열을 읽어오게 된다. 바이트열로부터 패킷의 시작점을 의미하는 싱크바이트(255,254 순으로 데이터가 배치되어 있다) 를 검출하는 "2. LXSDF T2 Tx 패킷 추출"에서 패킷 단위로 데이터를 분리해내고 패킷 내의 데이터 요소들을 "3. 패킷 데이터 파싱"에서 추출한다. 3 의 과정에서 확보된 각 데이터 요소들에 각각의 제품별로 전달되는 데이터의 내용은 다르다. 이들 정보들을 "4. 장치 제공 정보 확보" 단계에서 구하여 활용한다. 제품별로 다른 LXSDF T2 에 데이터 배치상황을 Device Specialization 이라고 부른다. 아래 그림의 2 번 단계와 3 번 단계의 코드 예를 다음 페이지에서 보이고 있다.

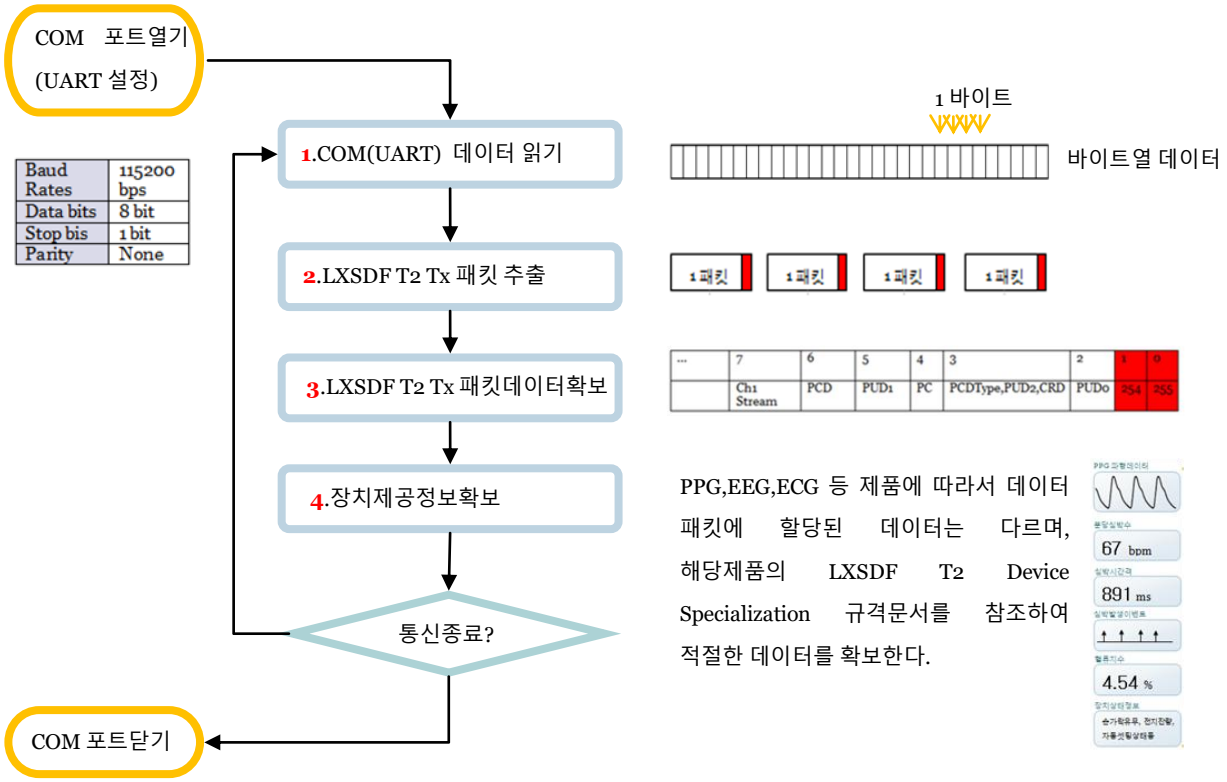


그림 2. 장치에서 송신된 LXSDF T2 Tx 를 수신하여 처리하는 호스트측 프로그램 전체 구조.

코드예 : LXSDF T2 Tx 패킷 추출 및 패킷 데이터 확보.

앞의 그림 2 의 단계 2 와 3 의 처리 코드 예를 보인다(C#코드). 간단한 일반 함수만 사용되고 있는것이라 언어와 무관하게 전체적인 코딩방법론은 동일하다.

```
// 시리얼로 수신한 데이터를 수신한 순서대로 1바이트 씩 함수 인자로 전달.
    // 함수에서 처리되는 것 : 싱크 지점 찾고 각 Packet_TX_Index 별로 데이터 추출한다.
    bool Sync_After = false;
    byte Packet_TX_Index = 0;
    byte Data_Prev = 0; // 직전값.

    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)// 싱크지점 찾았다.
      {
        Sync_After = true;
        Packet_TX_Index = 0;              // 패킷 TX인덱스 0으로 초기화.
      }

      Data_Prev = data_crnt;              // 현재 값을 직전 값으로 받아둔다.

      if (Sync_After == true)             // 싱크가 발견된 이후에만 실행된다.
      {
        Packet_TX_Index++;                // TX인덱스 1증가. 254가 발견된 지점이 1이다. 시리얼로 1바이트
수신될때마다 1씩 증가하는것.
        if (Packet_TX_Index > 1)          // TX인덱스 2이상만 수행된다.
        {
          if (Packet_TX_Index == 2)       // TX인덱스2 PUD0 확보.
            PUD0 = data_crnt;
          else if (Packet_TX_Index == 3)  // TX인덱스3 CRD, PUD2, PCD Type 확보.
            CRD_PUD2_PCDT = data_crnt;
          else if (Packet_TX_Index == 4)  // TX인덱스4  PC 확보.
            PacketCount = data_crnt;
          else if (Packet_TX_Index == 5)  // TX인덱스5  PUD1 확보.
            PUD1 = data_crnt;
          else if (Packet_TX_Index == 6)  // TX인덱스6 PCD(패킷순환데이터) 확보.
            PacketCyclicData = data_crnt;
          else if (Packet_TX_Index > 6)   // TX인덱스 7이상에는 스트림데이터(파형 데이터) 1바이트씩
순차적으로 들어온다. 데이터 수신되는 순서 -> 채널1의 상위바이트, 하위 바이트, 채널2의 상위바이트 하위바이트 ..
순서로 기록되어있다.
```

```
            {
                psd_idx = (byte)(Packet_TX_Index - 7); // PacketStreamData배열의 인덱스.
                PacketStreamData[psd_idx] = data_crnt; // crnt_data를 순차적으로 확보하여 스트림데이터만 확보한다.

                if (Packet_TX_Index == (Ch_Num * 2 * Sample_Num + 6)) // 채널수 x 2(2바이트 점유) x 샘플링 수량 +
6(파형데이터 구간 앞부분까지의 인덱스값) 까지가 1패킷의 끝이다.
                {
                    Sync_After = false; // 싱크지점 다시 검색되도록 false로 해둔다.
                    retv = 1; // 1패킷 단위의 파싱이 완료되면 리턴한다.
                }
            }
        } //if (Packet_TX_Index > 1)

    }
    return retv; // 1패킷이 완료되면 1을 반환, 그외에는 0반환.
}
```

표 5. 코드 예 : LXSDF T2 Tx 패킷 추출 및 패킷 데이터확보

Appendix A. COM 포트 자동 탐색법.

본 설명은 PC 등과 같이 장치가 어떤 COM 포트 번호할당이 고정적이지 않은 경우에 요구되는 장치 탐색에 관한 설명이다. Embedded System 에서 MCU 와 UART 접속하는 경우에는 자동탐색이 필요하지 않다.

개요.

호스트에서 COM 포트로 인식되는 장치인 경우, 응용프로그램에서 통신하기 위하여 사용자에게 COM 포트를 선택하라는 경우가 있다. 이는 사용자 편의성을 고려하지 않은 좋지 않은 제품 설계이다. 프로그램에서 자동으로 장치가 연결된 COM 포트를 찾는 방식으로 작동되도록 설계해야한다. 이 기능은 소프트웨어만으로는 해결될 수 없으며 장치에서 자동 탐색이 가능하게끔 기능구현이 되어있어야 한다.

COM 포트 탐색법.

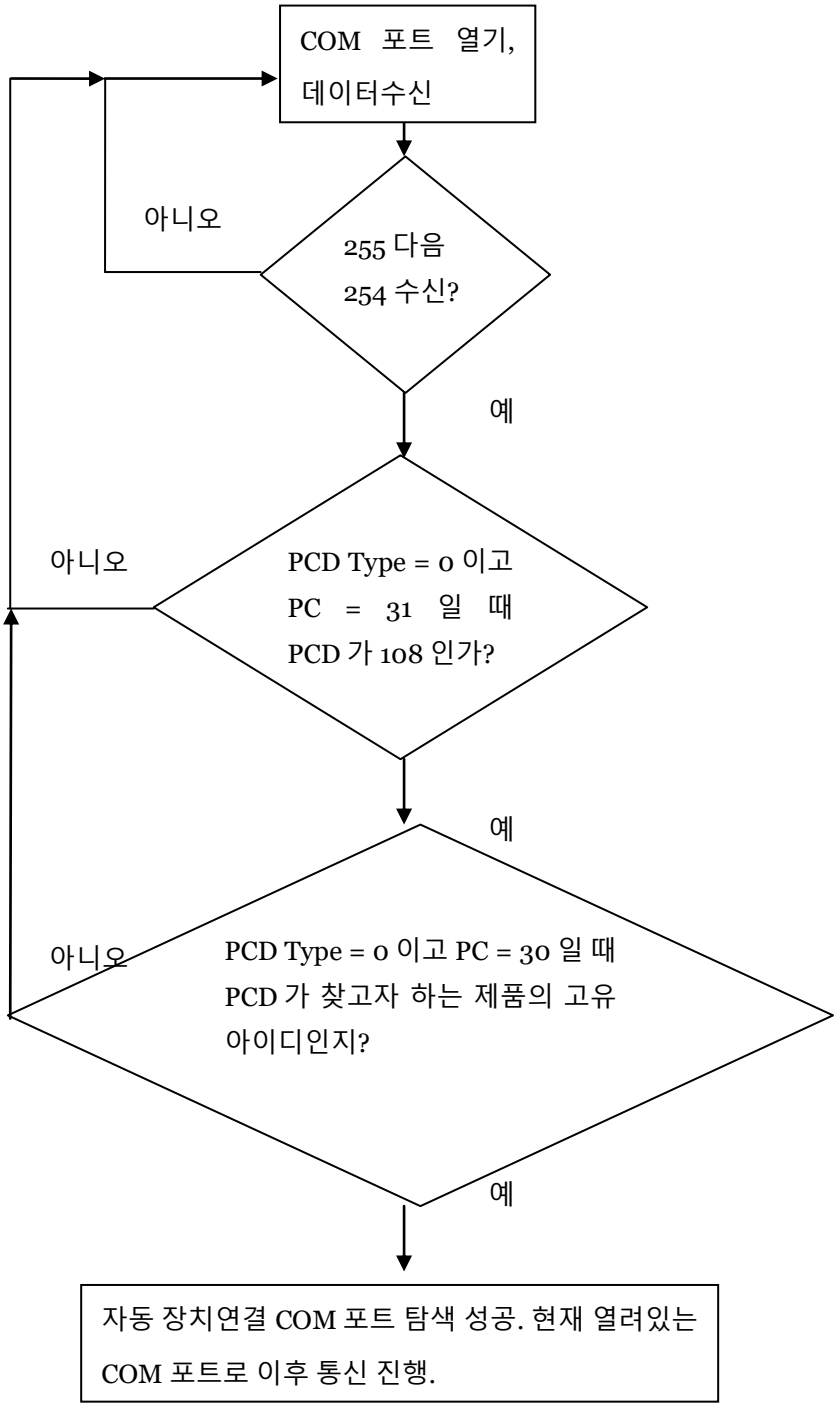
LXSDF T2 Tx 포맷에서는 장치 탐색에 활용하기 위한 정보로 PCD Type 0 이면서 PC=31 일때의 PCD 인 “COM 포트탐색정보”와 PC=30 일때의 PCD 인 “제품 고유 아이디”를 이용하여 원하는 제품에 해당하는 COM 포트를 자동으로 찾을 수 있다.

호스트에서 제공되고 있는 모든 COM 포트에 대해서 순차적으로 루틴을 돌면서 아래 설명의 과정을 거치면 목표하는 장치의 COM 포트를 쉽게 찾을 수 있다.

장치에 있는 COM 포트 1 개를 임시로 열어, 바이트 단위로 수신된 데이터를 읽어서 아래 표처럼 처리한다.

단계	설명.
단계 1	255 다음에 254 가 검출되었다면 -> 일단 통신가능한 장치후보이다. 255 다음에 254 데이터가 오지 않는다면 LXSDF T2 포맷은 아니다. 다른 COM 포트 열어서 다시 처음부터 시작.
단계 2	PC(패킷카운트) 값이 31 번까지 온다면 -> LXSDF T2 포맷으로 데이터를 전송해오는 장치임을 거의 확신하게된다. 그러나, 혹시나 다른 어떤 제품에서 우연의 일치로 잠시 LXSDF T2 와 동일한 포맷의 데이터가 전송될 수 있다. 안전하게 확정하기 위하여 PC = 31 에서의 패킷순환데이터 값을 확인하여 그 값이 108 이면 이것은 100% LXSDF T2 포맷임을 확정한다.
단계 3	LXSDF T 포맷으로 통신하는 장치임이 확인되었다면 그 다음 단계는 통신하고자 하는 제품 모델이 맞는지 검색한다. 이때는 패킷카운트 30 을 확인해서 통신의도하는 제품의 고유아이디인지 확인한다.

COM 포트 자동 탐색 흐름 예.



COM 포트 자동탐색 C# 코드 예.

C#이든, C++이든 언어무관하게 검색하고자 하는 장치의 자동탐색방법론은 동일함.

```
int bytestoread = sp.BytesToRead;   // com포트 버퍼에 있는 바이트 수 확보. Sp는 serial port object임.

// 판단1. 우리 장치인지 아닌지? 우리 장치는 COM포트에 데이터가 있어야만 한다.
if (bytestoread == 0) { return; }  // COM포트에 읽어들인 데이터가 없다는 것은 LXSDF T2 포맷은 아니다.
LXSDF T2는 항상 데이터를 전송하는 것으로 되어있다.

/// 최소한 COM포트에서 읽을데이터가 있다면 그 데이터를 모두 읽어들인다.
byte[] rbuf = new byte[bytestoread]; // 메모리 크기를 동적 생성했다.
bool find_sync = false;
sp.Read(rbuf, 0, bytestoread); // rbuf에 일단 받았다.
// 판단2. 싱크가 발견되는지 확인한다.
for (int i = 0; i < bytestoread-1; i++) //
{
   if (rbuf[i] == 255 && rbuf[i + 1] == 254) // 싱크 지점 찾았다.
   {
      find_sync = true;
      break; // 루프탈출
   }
}
if (find_sync == false) return; // 255,254순서의 데이터가 없다면 LXSDF T2는 아니다.
/// 판단3. 최소한 싱크가 발견되는 경우에는  패킷순환 데이터를 확인한다. 이를 확인하기위해서는
연속적으로 일정시간 이상의 데이터를 받아봐야만 한다.
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; // ComDeviceID는 1이상의 값만 할당 된다.
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); // 데이터 읽어오고 읽은 바이트 누적합 구하기 한다.

   sum_readbytenum += readbytenum;
```

```
            for (int i = 0; i < readbytenum-3; i++)
            {
               if (cbuf[i] == 255 && cbuf[i + 1] == 254) // 싱크지점 검출했다.
               {
                  Packet_Count = cbuf[i + 4];          // 패킷카운트값 확보했다.
                  PacketCyclicData = cbuf[i + 6];      // 패킷순환데이터 확보했다.

                  if (Packet_Count == 31 &&  PacketCyclicData == 108)// 패킷카운트가 31일때의 패킷순환 데이터가
108이면 LXSDF T2 타입임이 확실하다.
                     find_108 = true;
                  else if(Packet_Count == 30)                  // 제품고유 아이디  자리다.
                     find_ComDeviceID = PacketCyclicData;
                  else if (Packet_Count == 29)                 // 펌웨어 버전번호 자리다. UART로 펌업뎃이 이뤄질 경우에는
필수다.
                     find_firmversion = PacketCyclicData;
                  else if (Packet_Count == 28)                 // 스트림데이터로 전송되는 채널수.
                     find_NumChannel = PacketCyclicData;
                  else if (Packet_Count == 27)
                     find_NumSample = PacketCyclicData;

                  if (find_108 && find_NumSample > 0)  // for문 탈출 find_NumSample 은  패킷카운트 27에 있고,
find_108이 패킷카운트 31이므로 이것 2개가 찾아진것은 중간값들도 다 찾아진것이므로 더 볼것 없다. 루프탈출한다.
                  {
                     while_continue = false;
                     break;
                  }

               }
            }
            /// COM포트에서 수신한 데이터를 몇개까지 살펴볼것인지 최대값 지정해둔다. 이 값이 너무 큰 경우 장치
검색에 시간이 오래 소요되므로 가능한 짧게 잡는게 좋다.
            /// LXSDF T2형식에서 장치 탐색을 위하여 최소한 요구되는 데이터 수량은 최소한 32개 패킷을
받아봐야한다. 즉, 1패킷의 바이트수량 68바이트 x 32 = 2176 바이트이다. 3000 바이트 정도면 32패킷은 충분히
포함되어있어 장치탐색 정보를 확실하게 검사할 수 있다.
            /// 위 계산식에서 1패킷의 바이트 수량은 아래 8바이트 + 64바이트 로 구해진것이다.
            /// 8바이트 : 1패킷이 TX인덱스 0에서 6까지 8바이트
            /// 64바이트 :  스트림영역은 채널수 * 2(바이트) * 샘플수 인데 이는 제품마다 다른 값을 갖지만, LXSDF
T2에서의 할당하는 최대채널수는 8. 샘플수 4이내이므로, 최대 64바이트이다.
            /// x 32 : 패킷카운트 0~31까지를 수신해봐야 하므로, 32패킷은 받아봐야 한다.
            if (sum_readbytenum > 3000)  // 강제 루프 탈출 조건.
            {
               while_continue = false;
               break;
            }
         } // while
```

표 6. 코드 예 : COM 포트 자동탐색.

Revision History

Release Date	Doc. ID	Description of Change
2012-12-03	LXD12 V1.1	1. 표 레이블 추가. 표 목차 추가. 2. 표 4. PCD Type 0 의 PCD 지정 데이터에 ComPath 데이터 항목 및 Reserved 데이터 항목 추가. 페이지 12 3. LXSDF T2 의 Rx, Tx 구분 설명 및 그림 추가. 페이지 4
2017-02-14	LXD12 V2	url link modified.-