

# 1. Packet Format

The packet should be a binary blob, no encoding is necessary. The reason that encoding should be removed is that it results in a packet substantially larger than is necessary which in some operating systems results in dropped sections of packets.

We have found from various tests that we can reliably receive 31 bytes on most systems. The previous packet was 54 bytes, we lost half of most packets on some systems. This is not app related, several other users who have developed their own BLE implementations have confirmed this.

Therefore, we propose to make a shorter packet, to improve reception reliability on most systems.

The packet below is 19 bytes, which is substantially shorter than the previous format.

The actual data format is identical, except for the way serial number is formatted.

Each digit 0 – 9 of the serial number is stored separately.

			Bits								
C Struct	Byte #		b7	b6	b5	b4	b3	b2	b1	b0	
Start	0	Start COMMAND	0xF2								
Serial	1	Serial B3	Year ( 4 ... 0 )								
	2	Serial B2	Month (1 ... 0)				Serial Number Digit 4				
	3	Serial B1	Serial Number Digit 3				Serial Number Digit 2				
	4	Serial B0	Serial Number Digit 1				Serial Number Digit 0				
MainMode	5	MAIN LCD	MODE	0	0	0	0 ~ 24 (0x00 ~ 0x18)				
MainRange	6		RANGE	OFL	+/-	0	0	RANGE ( 0 ~ 6 )			
MainValue	7		Value_H	High Byte							
	8		Value_L	Low Byte							
SubMode	9	SUB LCD	MODE	100 ~ 199, 0 ~ 24							
SubRange	10		RANGE	OFL	+/-	k	Hz	0	Point( 0 ~ 4 )		
SubValue	11		Value_H	High Byte							
	12		Value_L	Low Byte							
BarStatus	13	BAR LCD	STATUS	0	0	0	USE	0~150	+/-	1000 / 500	
BarValue	14		VALUE	0	0	0	BAR GRAPH 0 ~ 25				
IconStatus1	15	ICON LCD	STATUS1	0	1KHz	1ms	DC + AC		AUTO	APO	BAT
IconStatus2	16		STATUS2	0	BT	⌞	REL	dBm	MIN/MAX		
IconStatus3	17		STATUS3	0	TEST	MEM		A-HOLD		AC	DC
Checksum	18	Checksum	XOR of bytes 0 ... 17								

An example of how to send the data can be seen on the next page.

Apart from encoding all the fields are identical to the implementation prior to this change.

The inactive sections (See page 4) of the packet should also be dropped.

## 2. Packet Data Structure

```
union Packet
{
    struct
    {
        u8      Start;

        // Serial Bytes (all 4)
        u32      Serial;

        //Main Bytes
        u8      MainMode,
                MainRange;
        u16      MainValue;

        //Sub Bytes
        u8      SubMode,
                SubRange;
        u16      SubValue;

        //Bargraph
        u8      BarStatus,
                BarValue;

        //Icons
        u8      IconStatus1,
                IconStatus2,
                IconStatus3;

        //XOR Bitwise checksum
        u8      Checksum;
    };
    u8 Bytes[19u];
};

void BLESendBytes(u8 * const pBytes, u16 const pCount)
{
    //Put your code here to send x bytes...
}

void SendPacket(Packet * const pInput)
{
    constexpr u8 bytes = 19u;
    u8 checksum = 0u;

    for (unsigned i = 0u; i < bytes; ++i)
        checksum ^= pInput->Bytes[i];

    pInput->Checksum = checksum;
    BLESendBytes(pInput->Bytes, bytes);
}
```

### 3. Usage Example

```
//To use it....
Packet data;

//Same format of the bytes as before except not
// Output as a binary blob not serial
data.Start      = 0xf2;
data.Serial     = 0xffff;

//Main LCD section
data.MainMode   = 0x12;
data.MainRange  = 0x12;
data.MainValue  = 1234;

//Sub LCD section
data.SubMode    = 0x12;
data.SubRange   = 0x12;
data.SubValue   = 1234;

//Bargraph LCD section
data.BarStatus  = 0x12;
data.BarValue   = 12;

//Icon LCD section
data.IconStatus1 = 0x12;
data.IconStatus2 = 0x12;
data.IconStatus3 = 0x12;

//Checksum is calculated in send function
SendPacket( &data );
```