LSW-1188-29-1F Communication Protocol

AS USED IN A GERMAN BRAND NAMED AFTER A JAPANESE DELICACY.

STEVE OSWALD

General

This document tries to describe the communication protocol used in the Lishui LSW-1188-29-1F e-bike controller. It can be used to build compatible displays.

Communication is done via UART with the following parameters:

> 9600 bit/s (bitrate)
> 8 bits per transfer
> 1 stop bit
> no parity bit

All knowledge found in this document comes from reverse-engineering and have not to be correct.

I'm not responsible for damages caused by using this document.

Display Side

The display is sending a request to the controller every 100 ms (default config). The request is exactly 9 bytes long and is build from 1 start byte, 5 data bytes, 2 bytes CRC and 1 end byte (return).

Byte	**Bitwise (if needed)**	**Description**	**Further information**
1	-	Start byte (0x55)	ASCII “U”
2	1	*Unknown (0)*	
	2	*Unknown (0)*	
	3	*Unknown (0)*	
	4	Light status (1 = on, 0 = off)	
	5	PAS level / Walk-assist (0x00 – 0x09, 0x0F)	The PAS level must be between 0 and 9 (even if there are only 3 or 5 PAS levels). While the value is 15 (0x0F) the walk-assist is active.
	6		
	7		
	8		
3	-	*Unknown (0x2D)*	
4	-	*Unknown (0x0F)*	
5	-	*Unknown (0x1B)*	
6	-	*Unknown (0x15)*	Maybe low voltage protection value.
7	-	CRC (low byte)	See “CRC Algorithm” Chapter
8	-	CRC (high byte)	
9	-	End byte (0x0D)	ASCII “\r”

Hints:

1. You can simply calculate the PAS level with the following algorithm:

```
if (PAS_LEVEL == 0) {
    return 0;
}

if (NUMBER_OF_PAS_LEVELS == 3) {
    return PAS_LEVEL * 3;
} else if (NUMBER_OF_PAS_LEVELS == 5) {
    return (PAS_LEVEL * 2) - 1;
}

return PAS_LEVEL;
```

Controller Side

The controller answers every request from the display. The answer is exactly 13 bytes long and is build from 1 start byte, 9 data bytes, 2 bytes CRC and 1 end byte (return).

Byte	Bitwise (if needed)	Description	Further information
1	-	Start byte (0x55)	ASCII "U"
2		Probably motor-watts	To get the actual watts, you must multiply with 10.
3	-	Wheel cycle time (in ms) (low byte)	Only the next package after one round contains a value. If you request a value during a round and you already got an value before you get " 0x00 ".
4	-	Wheel cycle time (in ms) (high byte)	
5	-	Error byte	See "Error Codes" Chapter
6	-	Status	0x02 = Ready 0x03 = Breaks are pulled 0x04 = Motor running
7	-	*Unknown (0x00)*	
8	-	*Unknown (0x00)*	
9	-	*Unknown (0x00)*	
10	-	*Unknown (0x00)*	
11	-	CRC (low byte)	See "CRC Algorithm" Chapter
12	-	CRC (high byte)	
13	-	End byte (0x0D)	ASCII "\r"

Hints:

1. As written in the further information you only get a wheel cycle time in the next request after the completed wheel cycle. This brings the problem, that you probably can't say for sure if this bike has stopped (value 0x00) or if this is a package during a round (value 0x00). Maybe best practice would be, to take the last value (for example 1720 ms), add 1000 ms and if you don't get a new value after this durations set the value to 0.

CRC Algorithm

The CRC is calculated by adding all previous bytes as low byte and high byte and simply output it as low byte and high byte.

Code example (C#):

```
private int FromHighByteLowByte(byte pHighByte, byte pLowByte) {
    int result = pHighByte << 8;
    result = result | pLowByte;

    return result;
}

private int GetCrc(byte[] pBytes) {
    int crc = 0;

    for (int x = 0; x < pBytes.Count; x = x + 2) {
        crc = crc + IntFromHighByteLowByte(pBytes[x + 1], pBytes[x]);
    }

    return crc;
}
```

Error Codes

There isn’t an official documentation and currently only one error code is known.

Error	**Description**
0x01	Communication Error (false CRC)

Communication Examples

Turning lights on at PAS level 3:

```
// Display -> Controller (lights off, PAS level 3)
0x55 0x03 0x2D 0x0F 0x1B 0x15 0x9D 0x25 0x0D
// Controller -> Display
0x55 0x00 0x00 0x00 0x00 0x02 0x00 0x00 0x00 0x00 0x55 0x02 0x0D

// Display -> Controller (lights on, PAS level 3)
0x55 0x13 0x2D 0x0F 0x1B 0x15 0x9D 0x35 0x0D
// Controller -> Display
0x55 0x00 0x00 0x00 0x00 0x02 0x00 0x00 0x00 0x00 0x55 0x02 0x0D
```

Turning on walk-assist at PAS level 6:

```
// Display -> Controller (walk-assist off, PAS level 6)
0x55 0x06 0x2D 0x0F 0x1B 0x15 0x9D 0x2A 0x0D
// Controller -> Display
0x55 0x00 0x00 0x00 0x00 0x02 0x00 0x00 0x00 0x00 0x55 0x02 0x0D

// Display -> Controller (walk-assist on, PAS level 6)
0x55 0x0F 0x2D 0x0F 0x1B 0x15 0x9D 0x33 0x0D
// Controller -> Display
0x55 0x00 0xB1 0x06 0x00 0x04 0x00 0x00 0x00 0x00 0x06 0x04 0x0D

// Display -> Controller (walk-assist on, PAS level 6) during wheel rnd
0x55 0x0F 0x2D 0x0F 0x1B 0x15 0x9D 0x33 0x0D
// Controller -> Display
0x55 0x00 0x00 0x00 0x00 0x04 0x00 0x00 0x00 0x00 0x55 0x04 0x0D

// Display -> Controller (walk-assist on, PAS level 6) after wheel round
0x55 0x0F 0x2D 0x0F 0x1B 0x15 0x9D 0x33 0x0D
// Controller -> Display
0x55 0x00 0xB1 0x06 0x00 0x04 0x00 0x00 0x00 0x00 0x06 0x04 0x0D
```