Flairics GmbH & Co. KG
Böhlerstraße 1
40667 Meerbusch
Germany

FLAIRICS SRXL Implementation

Revisions

Rev.	Datum	Author	Modification	Brief
1.00	28.02.2017	D.Picchi		1. Version, planning, definition

Content

INTRODUCTION **1**

HARDWARE **3**

FRAME SPECIFICATION **3**

FRAME **4**

CRC CALCULATION **5**

Introduction

- Definition based on the SRXL specifications Version 3.2:
  SRXL Specification V3_2.pdf
  SRXL Specification Version 3.2 / SRXL.org
  (Initial Draft 15 July 2010, Walter Meyer, last changes 24 March 2015, Walter Meyer)

Hardware

- 3pol. 100mil Plug
  - Pin1: GND
  - Pin2: RX (Data-Input; not used)
  - Pin3: TX (Data-Output)
- Low: <= 0.35V
- High: >= 2.0V
- Idle: High
- Setup: 115200Baud, 8 Databits, 1 Stop bit, none parity

Frame specification

- Frame-Intervall: 12ms

Frame

Byte: 1	2	3	4	5	6	7	8	9	10	...	n-3	n-2	n-1	n
Start	Version	Length	Status		Number of Servos	Servo 1		Servo 2		...	Servo x		CRC	
0xAE	0x10	n	MSB	LSB	x	MSB	LSB	MSB	LSB		MSB	LSB	MSB	LSB

Field	Byte	Range	**Description**
Start	1	Fix 0xAE	**Start byte for SRXL and Manufacturer-ID for Flairics GmbH & Co. KG**
Version	2	Fix 0x10	**In this document fixed as V1**
Length	3	0 ... 255	**Length of the message (n)**
Status	4 & 5	0x0000 ... 0xFFFF	**Status information** **Bit 7: RX is in Failsafe modus** **Bit 6: No servo values are sent from TX. RX-Unit gives no servo pulses yet.** **Bit 5: Binding is active** **Bit 4: RX is in Failsafe test modus** **Bit 3: reserved** **Bit 2: Battery warning** **Bit 1: Range Limit warning** **Bit 0: RX is paired with the TX-Unit**
Number Servos	6	0x00 ... 0x10	**Nr. of Servos in this Frame**
Servo	7 & 8 ff.	0x0000 ... 0xFFFF	**16 Bit Servo Value** **0xF800 (-2048) Servo value is not valid** **0xF801 (-2047) correspond to -100.0% 1000µs** **0xF802 (-2046)** **...** **0xFC00 (-1024) correspond to -50.0% 1250µs** **...** **0xFFFF (-1)** **0x0000 (0) correspond to 0% 1500µs** **0x0001 (+1)** **...** **0x0400 (+1024) correspond to +50.0% 1750µs** **...** **0x07FF (+2047) correspond to +100.0% 2000µs**
CRC	**n-1 & n**		**CRC lt. SRXL Specification**

CRC Calculation

- CRC Calculation according to ***universal_crc*** from Danjel McGougan: http://www.mcgougan.se/universal_crc/
- Calling Routine: ***universal_crc.exe -b 16 -p 0x1021 -a tab***

```
static const UINT16 CRC_LookUp[256] = {
          0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,  0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
          0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,  0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
          0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,  0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
          0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,  0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
          0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,  0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
          0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,  0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
          0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,  0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
          0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,  0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
          0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,  0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
          0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,  0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
          0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,  0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
          0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,  0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
          0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,  0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
          0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,  0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
          0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,  0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
          0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,  0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0
};

/*****************************************************************************//**
 * @brief             Berechnet SRXL CRC
 *
 * @param[in]         pData      - Bereich, über den die CRC berechnet werden soll
 * @param             Size       - Größe des Bereichs in Bytes (inklusive CRC)
 *
 * @return      CRC, (0 wenn CRC OK ist)
 */
UINT16     SRXL_CRC(UINT8 *pData, UINT32 Size)
{
UINT8      *p = pData;
UINT16     crc = 0;

          while (Size--)
                    crc = (crc << 8) ^ CRC_LookUp[((crc >> 8) & 0xFF) ^ *(p++)];

          return (crc & 0xFFFF);
}
/*****************************************************************************//**
 * @brief             hängt CRC an die letzten beiden Bytes von (Data+Size) an
 *
 * @param[in]         pData      - Bereich, über den die CRC berechnet werden soll
 * @param             Size       - Größe des Bereichs in Bytes (inklusive CRC)
 *
 * @return      CRC
 */
UINT16     SRXL_CRCAppend(UINT8 *pData, UINT32 Size)
{
UINT16     crc = SRXL_CRC(pData, Size - 2);

          pData += Size - 2;
          *(pData++) = crc >> 8;
          *pData = (crc & 0x00FF);

          return crc;
}
```