HORIZON HOBBY, LLC.

Specification for Spektrum® Bi-Directional SRXL

Based on the Multiplex v2.9

Rev B

2016 December 16

Specification for Spektrum® Bi-Directional SRXL

1 INTRODUCTION

This document is intended to serve as a Spektrum-specific guide to describe a bi-directional implementation of the SRXL protocol.

2 LEGAL INFO

This protocol, as described, in its entirety, without warranty or guarantee, can be freely and indefinitely, without limitation, implemented.

3 AUDIENCE

This document is intended for engineers to develop accessories that leverage the SRXL protocol. It assumes an intimate knowledge of how the Spektrum systems work.

4 RELATED DOCUMENTATION

All necessary technical information is contained within this document, including diagrams and source code guidance.

Multplex SRXL specification version 2.9 date 12 Feb 2014 by Walter Meyer.
Specification for Forward Programming, and
Specification for Spektrum Remote Receiver Interface

Telemetry Sensor Struct .h file
https://github.com/SpektrumFPV/SpektrumDocumentation/blob/master/Telemetry/spektrumTelemetrySensors.h

5 ELECTRICAL DATA

The protocol runs on servo-bus power (3.5V-8.4V). This allows us to extend the protocol into the servos themselves at a future date.

Signal is standard 3.3V logic with normal UART levels (3.3V = idle line), however the bus is generally in high-impedance state using the micro's internal pull-up. This is necessary for idle line detection, although it could leave us open to noise.

Devices on the bus may be powered by the bus, providing they are low-current. It would be unwise to run high-power servos on the same small-gauge wires.

Connection to low-power products is JST ZH4.

Pin W = Vbus (3.5-8.4V)

Pin X = Common/Ground/Return

Pin Y = signal (3.3V logic, high impedance)

Pin Z = N/C

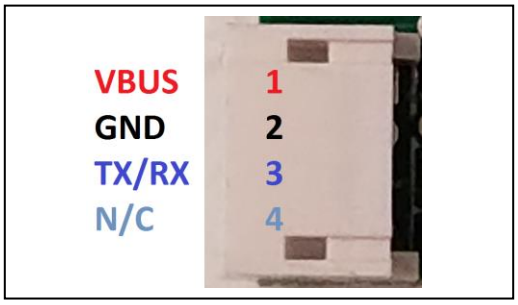

Photo 1: Labeled ZH4 Connector

6 HARDWARE-LEVEL PROTOCOL

Communications on the SRXL bus generally follow the MPX specification of 115,200 bps. Changes are:

a) The bus is not driven while idle.

b) The bus master shall be a serial remote receiver with telemetry ("telemetry remote"). It shall speak a single RF packet onto the bus every 11ms. The telemetry remote should require no changes other than a configuration parameter in order to support bi-directional SRXL.

c) Only one master is allowed on the bus.

d) Secondary remotes may be allowed, but shall follow the rules of other devices. These secondary remotes shall either be a telemetry remote operating in a secondary mode, or a device to interface a standard DSMX remote to the bi-directional bus.

e) Any non-master device can talk on the bus after a 1-character idle time.

f) Devices which spoke during the previous between-RF window shall remain silent (prevents hogging the bus).

g) All devices shall remain silent rather than write a message which would conflict with an incoming RF packet. You need to look ahead at your message to see if there is time for it, plus the 1-character idle time.

h) All devices shall listen for communications from the bus master (serial receiver) for synchronization.

i) In the case of a missing packet from the bus master, the bus shall remain idle for an additional 16ms in order to allow re-sync or new master selection.

j) New master selection is beyond the scope of this document at this time, but it shall be limited to other telemetry remotes.

k) Spektrum messages shall not exceed 64 bytes total, including header and CRC (this is to allow use of limited-size DMA buffer transfers)

7 SOFTWARE PROTOCOL

SRXL packets may be of different lengths. Packets of exactly 18 bytes which are identified as Spektrum data (type 0xA5) shall be RF data packets as currently generated by our products. All other packets shall have a different length, allowing differentiation.

Old 0xA2 packets shall be ignored. This protocol is only for products developed after May 2016.

All non-RF packets shall include an additional byte to serve as a sub-identifier or packet type immediately following the 0xA5 (manufacturer's data field, data 0). Packet types are:

Packet Type	ID	Length
Incoming RF Data	N/A	0x12
Telemetry Sensor Data	0x80	0x15
Bind Info	0x41	0x0F

Table 1: Packet Types

The Packet Type byte shall be immediately followed by an RFU byte.

7.1 Telemetry Sensor Data Packet

The Telemetry Sensor Data packet includes a complete 16-byte telemetry packet as it would be delivered over the RF as if it had come from an I^2C device on the X-Bus. This will enable use of existing telemetry devices by plugging them into an adapter interface (which would handle enumeration and SRXL protocol conversion), or by new sensors which utilize the SRXL protocol natively.

The packet format shall be:

<0xA5><0x80><Length><16-byte telemetry packet><CRC>

The <Length> shall be 0x15 (length of the 16-byte telemetry packet + overhead).

7.2 Bind Info Packet

Upon completion of bind the bus master shall issue a Bind Info packet. This packet could be used by other receivers to bind to the designated master without having had to listen to the bind over the air. The Bind Info packet can also be generated by a device on the bus to request that all listeners enter bind mode, or to query the bind status.

The packet format shall be:

<0xA5><0x41><Length><Request><GUID><Type><RF_chip_ID><CRC>

The <Length> shall be 0x13 (length of the payload + overhead).

<Request> is a single byte according to Table 2. Invalid values shall cause no action.

Request	Value
Enter Bind Mode	0xEB
Request Bind Status	0xB5
Bound Data Report	0xDB
Set Bind	0x5B

Table 2: Bind Request Options

<GUID> is the GUID used by the system, 64 bits.

<Type> is the bind type according to Table 3:

Bind Type	Value
NOT_BOUND	0xAE
DSM2 1024 22ms	0x01
DSM2 1024 (MC24)	0x02
DSM2 2048 11ms	0x12
DSMJ	0x92
DSMX 11ms	0xB2
DSM MARINE	0x50
DSMX 22ms	0xA2 (note conflict!)
DSMR 11ms or DSMR 22ms	0xA2 (note conflict!)
SURFACE DSM1	0x00
SURFACE DSM2 16.5ms	0x23
DSMR 5.5ms	(0xA4)

Table 3: Bind Status Types

Note that the current telemetry remote only supports non-surface bind types, so the conflict of DSMR and DSMX is immaterial. Also, only the command *Enter Bind Mode* is supported at this time.

When requesting Enter Bind, please refer to Table 4. These match the number of pulses normally sent by a base when putting a remote receiver into bind mode. When addressing all receivers (RF_chip_ID = 0, see below) it is recommended that you request type Bus Master DSMX/11ms (0x0A). A remote receiver SRXL interface (future product) would know to insert an extra pulse since legacy remotes would need to be in slave mode as they do not support telemetry.

Bind Request Type	Value
Bus Master DSM2/22ms	0x03
Bus Slave DSM2/22ms	0x04
Bus Master DSM2/11ms	0x05
Bus Slave DSM2/11ms	0x06
Bus Master DSMX/22ms	0x07
Bus Slave DSMX/22ms	0x08
Bus Master DSMX/11ms	0x09
Bus Slave DSMX/11ms	0x0A

Table 4: Maximum Bind Requests

<RF_chip_ID> is a (32-bit?) GUID value from the RF chip in the receiver. This is different than the GUID used for binding to the transmitter.

This value is 0 to address all receivers on the bus. When non-zero, only the device which matches that ID shall respond. In this manner a system could force all remotes into bind mode, or only a specific remote.

The Bound Data Report includes the ID of the reporting device. When a bind command was issued to

address 0, only the bus master shall report the status. When a bind command was issued to a specific receiver, that receiver shall issue the Bound Data Report.

REVISION HISTORY

Rev	Date	Author	Description
P0	2016-05-25	AK/TK/ MFA	For initial review.
P1	2016-07-20	MFA	Changed bind message length to 0x13. Added pinout labels for ZH4 connector
P2	2016-10-28	MFA/AK	Limit implementation status to *Enter Bind Mode* command.
A	2016-11-10	MFA	Typos, release.
B	2016-12-15	MFA	Added legal info