Specification for Spektrum Remote Receiver Interfacing

Enabling Use of Spektrum Remotes in Third-Party Products

Rev G

2019 January 22

Specification for Spektrum Remote Receiver Interfacing

1 INTRODUCTION

The market has a number of products which piggyback on Spektrum's proven reliable RF system by interfacing their flight control products directly to our remote receivers. The result is that many of these products do not properly implement the interface, relying on faulty assumptions and incomplete information as a result of the reverse engineering process.

2 AUDIENCE

This document is intended to assist hobbyists and flight control vendors to implement code and hardware which functions according to the protocol. A certain degree of electronics and firmware competency is necessary for proper implementation. The reader will need to self-determine their ability to properly implement the protocol.

3 RELATED DOCUMENTATION

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

4 LEGAL INFORMATION

This document does NOT contain any proprietary information. This document is subject to revision and does not constitute a commitment by Horizon Hobby to add support for any documented devices nor is it guaranteed to be error-free.

All correspondence regarding this document shall be through the Horizon Hobby legal department: Horizon Hobby, LLC, Legal Department, 4105 Fieldstone Road, Champaign, Illinois 61822 USA.

5 APPLICATION REQUIREMENTS

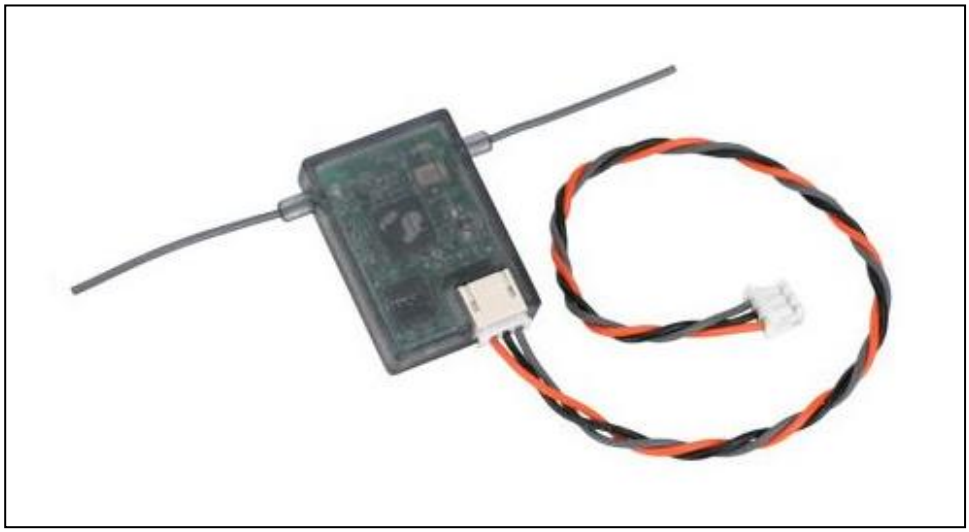
The requirements to create a robust RF link vary by application, installation, and flight range. In general longer range requirements (beyond park flying distances), and installations that contain substantial carbon fiber and metal components will require the use of more than one remote receiver oriented 90 degrees from each other. Preference should be given to airframe installation locations with line of sight access to the transmitter at all times (in all airframe orientations) and without direct interference from metal or carbon fiber components.

6 ELECTRICAL DATA

The interface to the remote receivers is a 3-wire bus. Pins are as noted in the photo below.

Pin	Color	Purpose
1	Orange	3.3VDC +/-5%, 20mA max
2	Black	GND
3	Gray	DATA

The connector used in the Spektrum remote is JST part number S3B-ZR(LF)(SN) (Digikey part 455-1670-ND). This mates to part JST ZHR-3 (Digikey 455-1160-ND). See JST for other mating components.



7 HARDWARE-LEVEL PROTOCOL

7.1 Normal Operation

In normal operation, the Spektrum remotes issue a 16-byte data packet every 11ms or 22ms, depending upon the selected protocol. The packet is transmitted at 115200bps, 8 bits, No parity, 1 stop (8N1). When no data is received, the remote does not transmit anything. It will count missed frames, and the updated count will be sent after the next packet has been received but not until.

If a failsafe operation is to be implemented, it must be triggered by repeated packet losses as timed by the flight controller. In Spektrum products this is a one-second timer which is reset upon the reception of any packet by the microcontroller. This provides one second of "hold last" signals. When the timer expires, the microcontroller executes the failsafe function defined for each channel.

7.2 Binding

A remote receiver is put into bind mode by issuing a series of pulses at power-up. The number of pulses determines the types of binds allowed, as well as the "internal" or "external" mode of operation (see below). The SPM9545 DSM2 remote does not accept DSMX bind commands. The DSMX remotes SPM9645 and SPM9646 support both DSMX and DSM2, and will automatically select DSM2 or DSMX as available in the transmitter. Also, as DSMX always uses 2048 mode, the increased precision gives the ability for an improved end-user experience. Note that DSM2 transmitters are not available in Europe, although users still operate them. For these reasons, we strongly recommend using only DSMX bind commands and DSMX remote receivers.

To put a receiver into bind mode, within 200ms of power application the host device needs to issue a series of falling pulses. The number of pulses issued selects which bind types will be accepted from the transmitter. Selecting the 11ms mode automatically allows either 11ms or 22ms protocols to be used. Selecting DSMX will automatically allow DSM2 to be used if the transmitter is unable to support DSMX. For this reason, we recommend that 11ms DSMX be used (9 ("internal") or 10

("external") pulses).

DSMX Bind Modes:

Pulses	Mode	Protocol Type
7	Internal	DSMx 22ms
8	External	DSMx 22ms
9	Internal	DSMx 11ms
10	External	DSMx 11ms

DSM2 Bind Modes (not recommended):

Pulses	Mode	Protocol Type
3	Internal	DSM2 22ms
4	External	DSM2 22ms
5	Internal	DSM2 11ms
6	External	DSM2 11ms

7.3 Internal and External Modes

In the Spektrum system, receivers bind as either a master (the "A" receiver) or auxiliary (the B, R, and L receivers). In most receivers the A is internal to the receiver body, hence the "internal" moniker. The important detail is that the A remote controls the bind process for the entire system, in that it is the only remote which negotiates with the transmitter to establish a link. For that reason, it is absolutely imperative that only one remote in your system be the "internal" or "A" remote. You may bind as many other receivers as desired at the same time, but they must all be bound in the "external" mode.

All remotes must be in bind mode for them to bind to the transmitter. Any remote not in bind mode will not bind.

If no remote is put into the internal mode, the transmitter will report Bind Failed and a link will not be established. If multiple remotes are in internal mode, the results cannot be predicted.

When a remote is put into bind mode, it retains its previously bound information until the bind is complete, at which time it saves the new bind information. For this reason, if you put a remote into bind mode and then cancel the bind (which can only be done by removing power), it will retain the information from the previous bind.

8 DATA FORMATS

Spektrum remote receivers output data in several different formats based upon protocol, speed, and internal/external status.

For internal remotes, the second byte (fieldname "system" in Section 8.3 below) indicates the protocol selected, which will also specify the data format used by for all remotes. Also for internal remotes, the first byte (fieldname "fades" in Section.3 below) indicates the fade count (missed frames for this remote). For external remotes, the first and second bytes are both used as a UINT16 to indicate fade count. This provides a higher fade cap than internal remotes. The remaining 14 bytes are the data packet.

For external remote, the first two bytes (fieldname "fades" in Section 8.4 below) indicate the fade count, and the remaining 14 bytes are the data packet.

Note that the internal remote provides the key for proper utilization of the data streaming from all remotes. All remotes are in 1024 mode, or all remotes are in 2048 mode.

The 14-byte data packet is either 1024 or 2048 servo data. DSM2/22 is the only 1024 packet; all others use 2048.

All data fields are big-endian, that is, the MSB is transmitted before the LSB. Bit 0 is the lsb, bit 15 is the msb.

8.1 Servo Field 1024 Mode

This format is used only by DSM2/22ms mode. All other modes use 2048 data.

Bits 15-10 Channel ID
Bits 9-0 Servo Position

```
#define MASK_1024_CHANID    0xFC00
#define MASK_1024_SXPOS     0x03FF
```

8.2 Servo Field 2048 Mode

This format is used by all protocols except DSM2/22ms mode.

Bits 15 Servo Phase
Bits 14-11 Channel ID
Bits 10-0 Servo Position

```
#define MASK_2048_CHANID    0x7800
#define MASK_2048_SXPOS     0x07FF
```

8.3 Message Structure Internal Remote

```
typedef stuct
{
        UINT8 fades;
        UINT8 system;
        UINT16 servo[7];
} INT_REMOTE_STR;
```

The “system” field will only contain certain values. Any value other than these should be ignored and the unit should behave in a mode appropriate for not having bound.

Allowed System Field Values

Value	Protocol
0x01	22MS 1024 DSM2
0x12	11MS 2048 DSM2
0xa2	22MS 2048 DSMX
0xb2	11MS 2048 DSMX

8.3.1 RSSI exception for SPM4649T

```
typedef stuct
{
```

```
        INT8 rssi;
        UINT8 system;
        UINT16 servo[7];
} INT_REMOTE_STR;
```

The 4649T with firmware version **1.1RC9 or later** will have RSSI in place of fades. This value is a signed integer represented in dBm units which will range from -42dBm (strongest signal) to -92dBm (weakest signal). If there's a connection/packet loss, the sent value will be -128. The host device could potentially have a startup value check to determine whether the receiver is sending fades or RSSI by checking whether the initial values received are negative or positive. It is up to the host device firmware developers to determine how they'd like to convert this value into a % that would be suitable to their devices intended application.

8.4 Message Structure External Remote

```
typedef struct
{
        UINT16 fades;
        UINT16 servo[7];
} EXT_REMOTE_STR;
```

8.5 Channel ID Information

The content of each 16-byte packet is determined by the transmitter, and may vary according to a number of factors such as bind type, aircraft type, and special modes such as X-Plus operation. In order to transmit all the channels possible, the transmitter makes determinations during operation which make it mandatory that the flight controller properly parse the channel data in each packet. One cannot assume that a packet will have the same data in the same index in the servo[] array in each frame; it is necessary that the Channel ID field be examined for each index in every packet.

Channel Identifiers

ID	Name
0	Throttle
1	Aileron
2	Elevator
3	Rudder
4	Gear
5	Aux 1
6	Aux 2
7	Aux 3
8	Aux 4
9	Aux 5
10	Aux 6
11	Aux 7

8.6 X-Plus Channels

X-Plus channels are encoded using a combination of the Phase bit, Channel ID, and re-assignment of two channel position bits to create an extended channel identifier. In particular, channels X+1 to X+4 are always found on Phase 0, and channels X+5 to X+8 are always found on Phase 1. The two highest bits of the Data are repurposed to provide the offset of 0 to 3 to the Channel ID of 12.

The sample code below will provide a quick conversion of a RF data with a specified phase into 11-bit channel position data. Note that this example is valid only for DSM2/11 and DSMX. X-Plus is only available in these protocols.

```
UINT16 servoPosition (  // Return servo pulse data
      UINT16 rfData,    // Raw data from the RF
      UINT8 phase,      // Phase bit (0/1)
      UINT8 *chan)      // Pointer to the channel number
{
      UINT16      sxPosition;

      *chan = rfData >> 11;
      *chan &= 0x0F;
      if (*chan < 12)               // Normal channels
      {
            sxPosition = (rfData & 0x07FF);
      }
      else if (*chan == 12)         // XPlus channels
      {
            *chan += ((rfData >> 9) & 0x03);
            if (phase)              // The phase is part of the X-Plus address
            {
                  *chan += 4;
            }
            sxPosition = (rfData & 0x01FF) << 2;
      }

      return (sxPosition);
}
```

9 Servo Position Values

To ensure consistency across implementation in multiple devices, the following interpretation of "Servo Position" data should be followed.

9.1 "Servo Position" Ranges

A full range of "Servo Position" data ranges from 0 to 1024 or 0 to 2048 depending on the bind type. These limits are equivalent to a ±150% travel setting in AirWare.

This full range translates to a range of 1194µs which when applied to a PWM servo signal equals 903µs to 2097µs

At ±100% travel, the data range is equivalent to a "Servo Position" data range of approximately 341 to 1707 which translated to PWM equals 1102µs to 1898 µs

9.2 1024 Mode Scaling

In a 1024 resolution link, the PWM pulse should be calculated as follows.

PWM_OUT = (ServoPosition x 1.166µs) + Offset

- PWM_OUT - the pulse length that is output to the servo or motor in µs.
- ServoPosition - satellite data received for a specific servo/motor which ranges from 0 to 2048
- 116.6 - a constant used to achieve a full pulse range of 1194µs
- Offset – offset to center the pulse at 1500µs. Theoretically, this should be 903µs, but the actual value to achieve a 1500µs

9.3 2048 Mode Scaling

In a 2048 resolution link, the PWM pulse should be calculated as follows.

PWM_OUT = (ServoPosition x 0.583µs) + Offset

- PWM_OUT - the pulse length that is output to the servo or motor in µs.
- ServoPosition - satellite data received for a specific servo/motor which ranges from 0 to 2048
- 58.3µs - a constant used to achieve a full pulse range of 1194µs
- Offset – offset to center the pulse at 1500µs. Theoretically, this should be 903µs, but the actual value to achieve a 1500µs

REVISION HISTORY

Rev	Date	Author	Description
P0	2015-05-07	AK	Initial authoring.
P1	2015-05-13	AK	Added image, corrected discrepancies vs. source documents.
P2	2015-05-20	VA	P1 revised and rewritten, P2 added 200ms
P3	2015-10-23	TK	Revised internal struct format and channel ID masks
P4	2015-10-28	MFA	Added Servo Position Values section.
A	2016-04-12	AK	Clean up for publication.
B	2016-04-15	AK	Correct typo
C	2016-12-16	MFA	Fixed description for system/fade byte positions
D	2017-12-19	MFA	Added struct for 4649T RSSI output
E	2018-01-30	MFA	Change baud rate to 115200bps
F	2018-07-17	AK	Section 8.6 – X-Plus channel decoding
G	2019-01-22	MFA	Corrected PWM scaling decimal position. Section 9.