Kambalunga's Channel Decoder/Expander for Spektrum SRXL2 Receivers

Spektrum's latest generation of receivers (the "SRXL2" receivers) are each capable of providing a serial output containing channel information from ALL transmitted channels regardless of how many physical servo output ports are available on the receiver. Thus, even though the AR637T receiver only has 6 servo output ports, additional channel information is received and may be used internally (such as for AS3X or SAFE controls). It is also made available in the serial SRXL2 output and may be accessed for controlling additional servos or other devices such as lights, retracts, etc., with appropriate "expander" hardware.

RCGroups.com contributor Kambalunga posted Arduino code for a channel decoder/expander for Spektrum's recent SRXL2 receivers. This will take the SRXL2 signal available from receivers such as the 6 channel AR630, AR631, AR6610T and AR 637T, the 8 channel AR8020T and AR8360T, or the remote SRXL2 serial receivers such as SPM9747, decode the channel data, and provide conventional PWM "servo" outputs to for up to 12 channels (determined by the number of channels your transmitter is capable of sending). So, for example, if you have a plane with an AR637T using throttle, ailerons, elevator, rudder, retracts and flaps (6 channels) and would like an additional channel or two for nosewheel steering, bomb drop, air brake, etc., Kambalunga's expander can provide those additional outputs without having to buy a new receiver.

Other than connectors, only two components are needed – an appropriate Arduino microcontroller and a Schottky diode. These are available very inexpensively – my implementation using an Arduino Nano clone came to under $6 excluding the connectors. I have tested my device with several different receivers in a number of different configurations and have found that it operates very effectively.

Kambalunga's description of his design and code is very brief and somewhat cryptic. With his permission this article provides an expanded explanation, and description of how I implemented it. It is not a detailed construction article, however. The wiring details are up to you.

Spektrum Servo and Serial Port Nomenclature

On the SRXL2 receivers the servo port slots are identified by numbers, starting at 1 (conventionally the throttle). In ascending order the others then are customarily assigned to 2 = aileron, 3 = elevator, 4 = rudder, 5 = gear, 6 = flap (AUX1), etc.

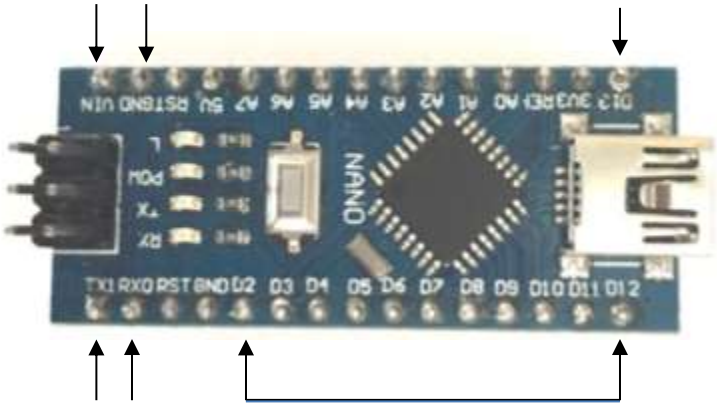
To the left of Slot 1 is a slot marked "Bind/Prog/SRXL2" (BPS for brevity) which accepts a standard servo connector. On ***some of*** the receivers there is also a 4 pin port on top marked "SRXL2" which typically is used for remote receivers or additional telemetry modules. The BPS port and the 4 pin port are electrically connected and logically the same so cannot be used independently for SRXL2 purposes. See the following photo of the AR8020T receiver showing the slot numbering

In addition there is also an SRXL2 port shared with (Throttle) Port 1. The SRXL2 line to the BPS port and to Port 1 are independent so can be used separately without interfering with each other. On the BPS port or Port 1 the SRXL2 signal line is the yellow or white wire on a regular servo connector and the others are the + and – power wires.
If you are using a SMART ESC then Port 1 is pre-empted by SMART.

The overall consequence of all this is that you can get an SRXL2 signal from the top port (if it exists), the BPS port, or Port 1, but ***only two can be used at the same time***. If you have a remote RX in the top port, get your SRXL2 signal from Port 1. If you have a SMART ESC in Port 1 get your SRXL2 signal from the top port or the BPS port. If you have both a SMART ESC and a remote RX, you are out of luck with the expander.

Arduino Nano Inputs

The Arduino takes its serial input/output from 2 ports marked TX1 and RX0 in the pinout below. It needs both input and output to communicate with the SRXL2 device. Since the SRXL2 only uses one line for both input and output, a method to separate these has to be provided to avoid conflicts. This is provided by connecting the SRXL2 line from the receiver directly to the RX0 pin, but by way of a Schottky diode to the TX1 pin. I used an inexpensive 1N5711 diode ($5 for 10 from Amazon, even cheaper elsewhere). The cathode of the diode (the end with the white band) goes to the TX1 pin. The other end of the diode is connected to the SRXL2 line from the receiver.



Power to the Arduino is provided at the VIN (+5V) and GND pins (marked by arrows in the pinout). These can come from the receiver SRXL2 connector too. The PWM signal outputs come from pins D2 through D13 and will drive servos directly (D13 is on the opposite side of the board from the other outputs). D2 is the throttle channel, D3 is aileron, D4 is elevator, etc. D8 will be AUX2, D9 = AUX3, etc.

There is no need to duplicate ports already available on your receiver (aileron, elevator. . .) **BUT if you are running a remote receiver and have to use Port 1 for the expander, then you need to get your throttle signal from pin D2 instead of from the receiver.**

Programming the Arduino

The Arduino Nano needs either the Arduino IDE to be downloaded to your PC, or needs the online Arduino Create running on a web browser. You can find these using Google. Uploading Kambalunga's sketch to the Nano is straightforward as long as you have the correct drivers loaded.

Kambalunga's sketch is at

https://www.rcgroups.com/forums/showatt.php?attachmentid=14420391&d=1608241595

If you read through his code you will notice that being a computer person, he starts numbering at ZERO rather than at one (0 = throttle, 1 = aileron, etc).

Finally

If none of the above makes much sense to you, I suggest you don't tackle this project without help from someone with electronics experience. It is not a difficult project but is not recommended for a first electronic project.

I am providing this information free of charge and with no warranty for fitness of purpose, and the customer service you can expect is exactly what you paid for.

Acknowledgement: The entire design of the expander/decoder is due to Kambalunga, who also answered a bunch of my questions about how to get it up and running.

kallend@iit.edu