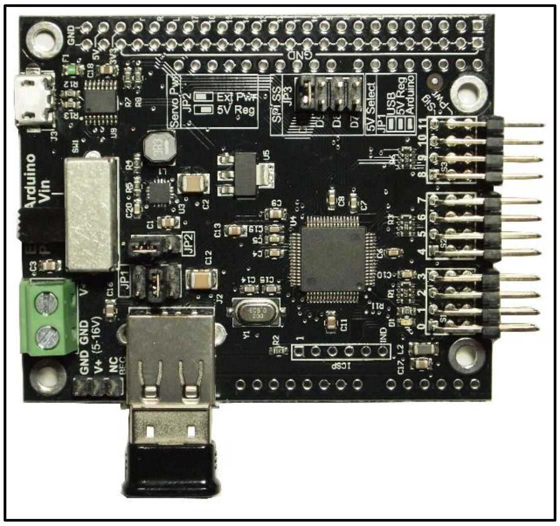
SERVOSHOCK2 SERVO AND I/O CONTROLLER



The ServoShock is an RC servo and digital output controller for the following input devices:

- Sony DualShock4 controller
- Thrustmaster T.Flight Hotas 4 for PS4
- 3DConnexion SpaceNavigator
- 3DConnexion SpaceMouse Wireless
- 3DConnexion SpaceExplorer

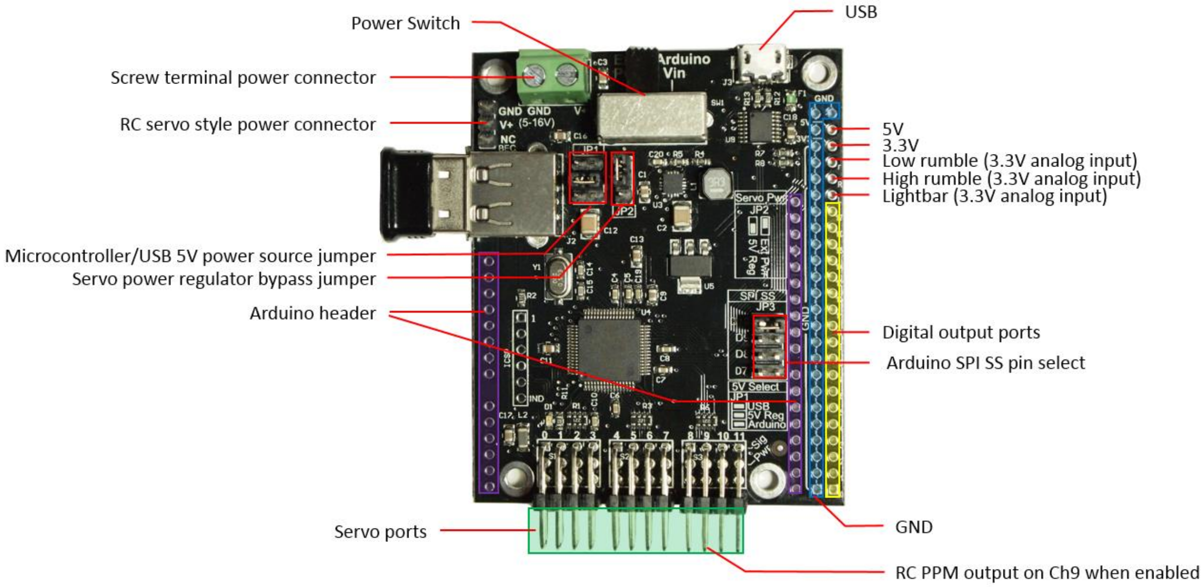
Features:

- **10 meter wireless range**
- **12 RC Servo outputs**: RC servos are controlled by joysticks, triggers, touchpad, and x/y tilt.
- **18 digital outputs**: Controller buttons and triggers control digital I/O pins. Five different modes are available for each button: pushbutton, toggle, single-shot, autofire, and toggle autofire.
- **Output adjustment/configuration**: Both servo and digital I/O outputs have several adjustment and configuration parameters that can be set using only the controller.
- **RC radio interface**: The 12 servo channels can be output to the XJT FrSKY transmitter.
- **Differential/Mecanum wheel signal mixer**: The joystick signal output signals can be mixed for controller differential drive or mecanum wheel platforms.
- **Control of DualShock LEDs and rumble motors**: The lightbar and rumble motors in the DualShock controller can be set using analog input pins or the SPI interface.
- **SPI bus data and control interface**: A SPI bus interface is provided to enable the user to read data packets from the DualShock controller and manually control the ServoShock outputs or set the DualShock's LEDs and rumble motors. All four SPI bus modes can be used up to 1MHz clock rate.
- **Arduino Library**: The ServoShock can be used as a stand-alone controller or as an Arduino shield using the Arduino library.

Contents

1.0 Getting Started

1.1 Overview


Power Switch
USB
Screw terminal power connector
RC servo style power connector
5V
3.3V
Low rumble (3.3V analog input)
High rumble (3.3V analog input)
Lightbar (3.3V analog input)
Microcontroller/USB 5V power source jumper
Servo power regulator bypass jumper
Arduino header
Digital output ports
Arduino SPI SS pin select
Servo ports
GND
RC PPM output on Ch9 when enabled

1.2 Input Power

The recommended way to power the ServoShock is to use a BEC for RC receivers (such as the Castle Creations 10A BEC), and set jumper JP2 to the 5V regulator bypass (top) position. This will connect the servo power line directly to the input power source. **Warning: When using this configuration, do not connect a battery directly to the ServoShock without an in-line regulator or BEC, or you may over-volt your servos.**

If your servos do not use much power, you may be able to share the ServoShock's logic power regulator (set jumper JP2 to bottom position); however, this may cause the logic power to brown out if the servo draws too much power.

Connect the servo power supply to the screw terminals or the RC style connector. The maximum voltage is 16V when using the on-board 5V regulator to power the servos, or the maximum servo voltage if you are bypassing the 5V regulator.

1.3 Jumper Settings

JP1

Top Position: 5V logic gets its power from the USB micro connector.

Middle Position: 5V logic power comes from the on-board regulator which steps down external power to 5V.

Bottom Position: 5V logic power comes from the Arduino.

JP2

Top Position: Servos powered directly from the external power input.

Bottom Position: Servos powered from 5V logic power.

JP3

Arduino UNO SPI bus slave select: D7, D8, D9, or D10.

1.4 Connecting Servos

Plug the servos into the header pins. The ground connection is at the bottom.

1.5 Output Mapping

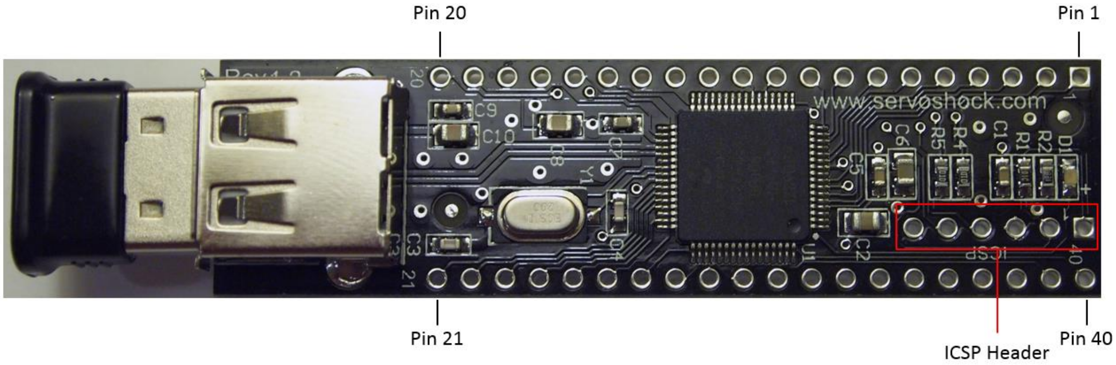
Dualshock 4	T.Flight Hotas 4 For PS4	SpaceExplorer/ SpaceNavigator/ SpaceMouse Wireless	**ServoShock Servo Channel**	**Config Menu Color Code (See section 3.0)**
Left Joystick X	X-axis	Translate X	**0**	**Red**
Left Joystick Y	Y-axis	Translate Y	**1**	**Orange**
Right Joystick X	Z-axis	Translate Z	**2**	**Yellow**
Right Joystick Y	Throttle	Roll X	**3**	**Lime**
Left Trigger	Throttle paddle	Roll Y	**4**	**Green**
Right Trigger	R2/L2	Roll Z	**5**	**Turquoise**
Touchpad Left X	Hat Left/Right	Shift/Ctrl*	**6**	**Blue**
Touchpad Left Y	Hat Up/Down	Esc/Alt*	**7**	**Purple**
Touchpad Right X	Circle/Square	R/L*	**8**	**Pink**
Touchpad Right Y	Triangle/Cross	T/F*	**9**	**White**
Tilt X	R1/R3	1/2	**10**	**Red Fast Blink**
Tilt Y	L1/L3	+/-*	**11**	**Orange Fast Blink**
Button	Button	Button	**Digital Output Channel**	
Share	Share	Panel*	**0**	**Yellow Fast Blink**
Left Joystick Button	L3	+*	**1**	**Lime Fast Blink**
Right Joystick Button	R3	-*	**2**	**Green Fast Blink**

Options	Options	Fit*	**3**	**Turquoise Fast Blink**
D-Pad Up	Hat Up	Esc*	**4**	**Blue Fast Blink**
D-Pad Right	Hat Right	Ctrl*	**5**	**Purple Fast Blink**
D-Pad Down	Hat Down	Alt*	**6**	**Pink Fast Blink**
D-Pad Left	Hat Left	Shift*	**7**	**White Fast Blink**
Left Trigger	L2		**8**	**Red Slow Blink**
Right Trigger	R2		**9**	**Orange Slow Blink**
Left Bumper	L1	1	**10**	**Yellow Slow Blink**
Right Bumper	R1	2	**11**	**Lime Slow Blink**
Triangle	Triangle	T*	**12**	**Green Slow Blink**
Circle	Circle	R*	**13**	**Turquoise Slow Blink**
Cross	Cross	F*	**14**	**Blue Slow Blink**
Square	Square	L*	**15**	**Purple Slow Blink**
PS Button	PS Button	2D*	**16**	**Pink Slow Blink**
Touchpad Button			**17**	**White Slow Blink**

*SpaceExplorer Only

1.6 Pinout for DIP-40 Module

The ServoShock is also available in DIP-40 module that makes it easy to integrate into your own projects. 5V and 3.3V must be supplied to the DIP-40 module.



ServoShock DIP-40 Module

PCB Pin Number	Channel Function	Pin Type
1	Left stick X-axis	Servo Output
2	Left stick Y-axis	Servo Output
3	Right stick X-axis	Servo Output
4	Right stick Y-axis	Servo Output
5	Left trigger analog position	Servo Output

6	Right trigger analog position	Servo Output
7	Touchpad Left X	Servo Output
8	Touchpad Left Y	Servo Output
9	Touchpad Right X	Servo Output
10	Touchpad Right Y	Servo Output
11	Slow rumble strength	Analog Input
12	Fast rumble strength	Analog Input
13	Tilt X	Servo Input
14	Tilt Y	Servo Input
15	UART-Tx	UART Output
16	UART-Rx	UART Input
17	3.3 V	Power
18	GND	Ground
19	5 V	Power
20	Select	Digital Output
21	SCK	SPI Clock Input
22	MOSI	SPI Data Input
23	MISO	SPI Data Output
24	/SS	Slave Select Input
25	Left stick button	Digital Output
26	Right stick button	Digital Output
27	Start	Digital Output
28	D-pad up	Digital Output
29	D-pad right	Digital Output
30	D-pad down	Digital Output
31	D-pad left	Digital Output
32	Left trigger on/off	Digital Output
33	Right trigger on/off	Digital Output
34	Left bumper	Digital Output
35	Right bumper	Digital Output
36	Triangle	Digital Output
37	Circle	Digital Output
38	Cross	Digital Output
39	Square	Digital Output
40	PS button	Digital Output
ICSP 1	Reset	Digital Input
ICSP 2	3.3V	Power
ICSP 3	GND	Ground
ICSP 4	PGD	Programmer Data
ICSP 5	PGC	Programmer Clock
ICSP 6	NC	Not Connected

*The LED color control pin and the Touchpad button output are not available on the DIP-40 module.

1.7 Pairing the Controller for Bluetooth

To pair the controller, perform the following steps:

1. Insert the Bluetooth dongle into the ServoShock's USB port.
2. Power on the ServoShock for at least 3 seconds, and then remove the Bluetooth dongle.
3. Plug in the DualShock4 controller. If the controller is successfully recognized, the outputs can be controlled over the USB link.
4. Remove the DualShock4's USB cable, and replace the Bluetooth dongle.

1.8 Connecting the Controller

To connect the controller via wired USB, plug the controller into the USB port.
To connect the controller via Bluetooth, press the **PS** button on the DualShock4 after pairing.

1.9 Disconnecting the Controller

To disconnect the controller, either

- hold down the **PS** button for 10 seconds
- hold down **Share+Options** for 3 seconds

When disconnected, the ServoShock outputs will revert to a neutral state (as if the user had let go of the controller). Toggled outputs will remain toggled.

1.10 Checking the Controller Battery Level

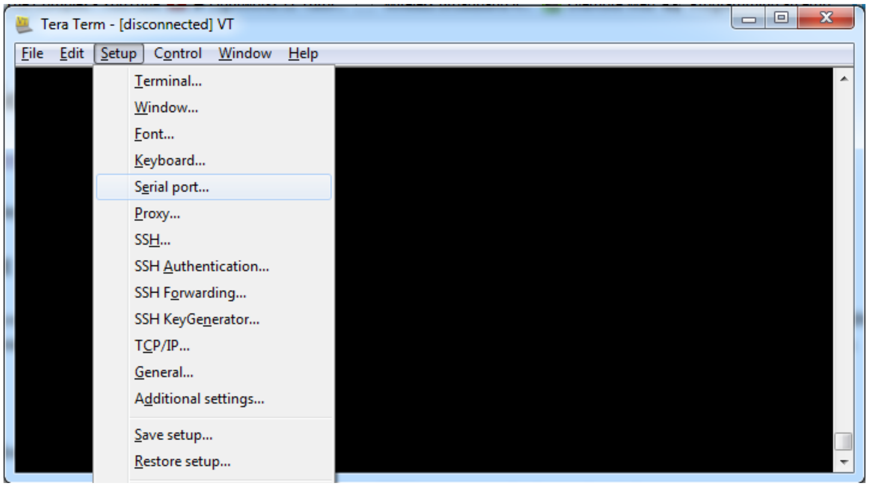
By default, the lightbar on the controller will indicate the controller’s battery level (green, yellow, red). When the lightbar color is set to a specific color, to check the controller battery level, press the **PS** button (don't hold it down for more than 10 seconds, or the controller will disconnect). The battery level will be temporarily displayed on the controller lightbar.

2.0 USING THE SERIAL READOUT

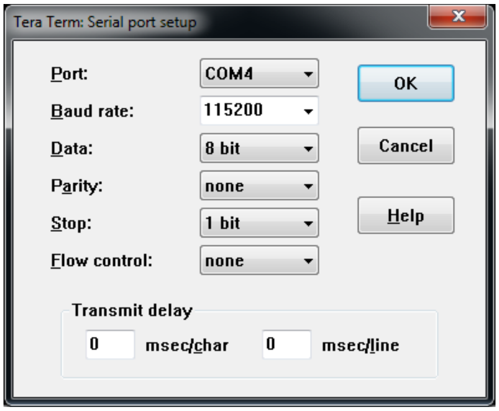
Using the serial port will make it much easier to configure the controller. To use the serial readout with the ServoShock shield board, plug the shield board into a USB port on your computer via the mini-USB port next to the power switch. If the driver for the FTDI USB-to-Serial chip are not automatically installed, they can be downloaded at the FTDI or ServoShock website. If the shield board is not used, then you must provide your own RS-232/UART solution and interface with pins 15 and 16 of the module.

2.1 Using Tera Term

To open the serial port on a computer, download and install Tera Term (http://ttssh2.sourceforge.jp/). Start Tera Term, and go to **Setup->Serial Port**.



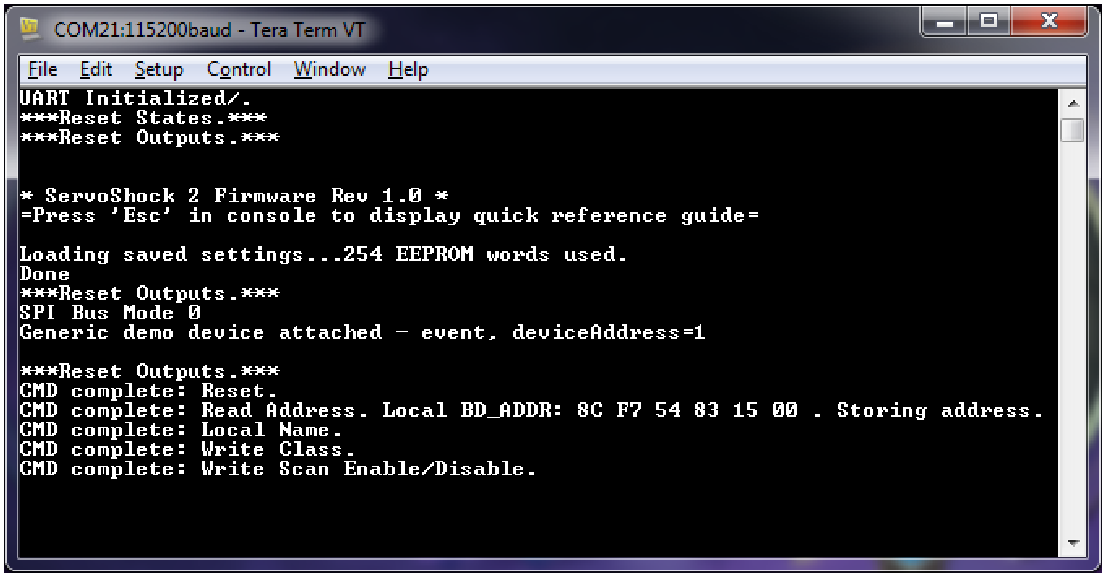
Then, select the serial port that corresponds with the ServoShock. (If you don't know which port is the correct port, you may have to try each of them until you find the right one.) Use these settings:



Port: COM4
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none
Transmit delay
0 msec/char 0 msec/line

Hit **OK** to open the port.

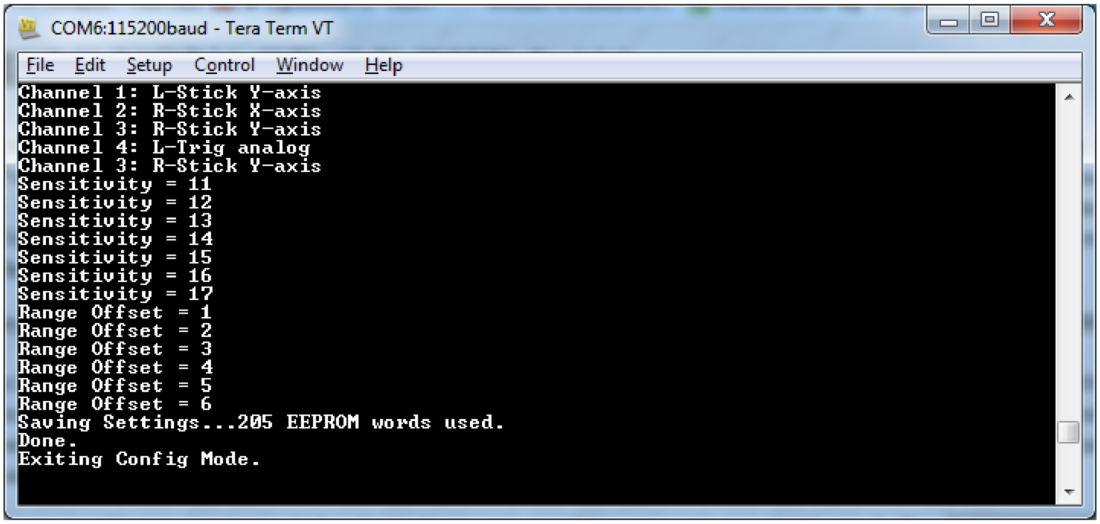
If you've selected the correct port and set it up correctly, unplugging and re-plugging the USB dongle should results in a readout like the following:



```
UART Initialized/.
***Reset States.***
***Reset Outputs.***


* ServoShock 2 Firmware Rev 1.0 *
=Press 'Esc' in console to display quick reference guide=

Loading saved settings...254 EEPROM words used.
Done
***Reset Outputs.***
SPI Bus Mode 0
Generic demo device attached - event, deviceAddress=1

***Reset Outputs.***
CMD complete: Reset.
CMD complete: Read Address. Local BD_ADDR: 8C F7 54 83 15 00 . Storing address.
CMD complete: Local Name.
CMD complete: Write Class.
CMD complete: Write Scan Enable/Disable.
```

Tip: In the terminal, you can hit ESC at any time to print display a quick reference guide.

Now, connect the controller and follow the directions in Section *3.0 Configuration Settings* to configure the controller. During configuration, changes in the settings will be displayed in the terminal as seen below:



```
Channel 1: L-Stick Y-axis
Channel 2: R-Stick X-axis
Channel 3: R-Stick Y-axis
Channel 4: L-Trig analog
Channel 3: R-Stick Y-axis
Sensitivity = 11
Sensitivity = 12
Sensitivity = 13
Sensitivity = 14
Sensitivity = 15
Sensitivity = 16
Sensitivity = 17
Range Offset = 1
Range Offset = 2
Range Offset = 3
Range Offset = 4
Range Offset = 5
Range Offset = 6
Saving Settings...205 EEPROM words used.
Done.
Exiting Config Mode.
```

3.0 CONFIGURATION SETTINGS

To change the configuration of the ServoShock, you must enter configuration mode by pressing and holding down the **PS** + **Share** buttons for 3 seconds. While in this mode, you can cycle through each of the outputs channels with **Left Bumper** or **Right Bumper** in order to adjust the settings for that particular output. See Section 1.5 Output Mapping for channel color codes. To save and exit the configuration mode, press and hold the **Options** button for 3 seconds. To discard the settings, power cycle the ServoShock module.

During configuration, the changed values can be read out from the UART, and the lightbar will display a color code that corresponds with the channel.

3.1 General Commands

3.1.1 Enter configuration mode

Hold Share + PS Button (3 seconds)

Hold down **Share+PS** for 3 seconds, release when the controller rumbles.
This command will put the ServoShock in configuration mode.

3.1.2 Save and Exit

Hold Options (1 second)

Hold down **Options** for 1 second, release when the controller rumbles.
This command will commit the settings to non-volatile memory and exit the configuration mode.

3.1.3 Load defaults

Hold Share (1 second / 3 seconds)

Press and hold **Share** for 1 second to load the default settings for the current channel, and release when the controller rumbles.
To load defaults for all channels at once, hold down **Share** for 3 seconds, and release when the controller rumbles twice (first a short rumble followed by a long rumble).
This command will load the default settings, but does not save them until the “save and exit“ command (3.1.2) is executed.

3.1.4 Next/Previous Output Channels

Right Bumper/Left Bumper

Use **Right Bumper** and **Left Bumper** buttons to cycle forward and back through the output channels in configuration mode.

3.1.5 Cycle SPI Bus Modes

R-Stick + Triangle

Hold **R-Stick** and press **Triangle** to cycle through the SPI bus modes. Four modes are available:

- Mode 0: Polarity = 0, Phase = 0 (default)

- Mode 1: Polarity = 0, Phase = 1
- Mode 2: Polarity = 1, Phase = 0
- Mode 3: Polarity = 1, Phase = 1

3.1.6 Enable/Disable Idle Disconnect

R-Stick + Cross

- Enable auto-disconnect (default)
- Disable auto-disconnect

Hold **R-Stick** and press **Cross** to enable or disable the 10-minute idle auto-disconnect power saving feature.

3.1.7 Disconnect Behavior for Toggled Digital Outputs

R-Stick + PS Button

- Preserve output states (default)
- Reset output states

This setting determines if the toggled outputs are preserved or reset when the controller is disconnected.

Hold **R-Stick** and press the **PS Button** to cycle between the modes.

3.1.8 LED Brightness

L-Stick + D-Pad Left/Right

- Default setting: 1
- Lowest setting: 0 (Lightbar off)
- Highest setting: 10

This setting controls the brightness of the lightbar. Hold **L-Stick** and press the **D-Pad Up/Down** buttons adjust the brightness.

3.1.9 LED Color

L-Stick + Share

- Red
- Orange
- Yellow
- Lime
- Green
- Turquoise
- Blue
- Purple
- Pink
- White

- Color set by analog pin
- Green/Yellow/Red battery level indicator(Default)

Press **L-Stick** + **Share** to cycle through various colors of the lightbar. By default, the lightbar acts as a battery life indicator. When the lightbar is set to a different color, the battery level can be checked by pressing the **PS** button.

3.1.10 PPM Output Mode

L-Stick + Options

- RC transmitter PPM signal disabled (default)
- RC transmitter PPM signal output on servo channel 9

The ServoShock can output a PPM signal compatible with the FrSKY XJT transmitter module on servo channel 9. Press L-Stick + Options to toggle this option on and off. The XJT module uses 6-15V for power, so you should set JP2 should be set to the top *Ext Pwr* position. Do not plug other servos into the ServoShock unless you are sure they are compatible with the input voltage. See section **4.0 FrSKY XJT Radio Module Adapter** for details on using the radio module adapter board.

3.2 Servo Adjustments

The following commands in this section are valid when configuring channels 0 through 11 (servo channels).

3.2.1 Position/Incremental Mode

R-Stick + Right Bumper

- Position Mode (Default)
- Incremental Mode

Hold **R-Stick** and press **Right Bumper** to toggle between position and incremental modes. In position mode, the analog input changes the position of the servo. In incremental mode, the analog input controls the velocity of the servo.

3.2.2 Invert Direction

R-Stick + Left Bumper

Hold **R-Stick** and press **Left Bumper** button to invert the movement direction of the servo.

3.2.3 Adjust sensitivity

Triangle/Cross

Press the **Triangle/Cross** button to increase/decrease analog input sensitivity.

- Default: 10
- Min: 1
- Max: 200

3.2.4 Adjust Center Deadband

Square/Circle

- Default: 0
- Min: 0
- Max: 125

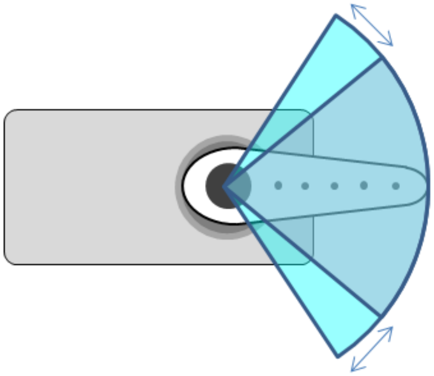
Press the **Circle** /**Square** button to increase/decrease the center “zero” region for the analog inputs, such as the joysticks and accelerometers. By default, the deadband is set to 0. Increasing the deadband may require the user to increase the sensitivity of the input to be able to achieve the servo’s full range of motion, since the deadband reduces the available input value range.

3.2.5 Range of Motion

D-Pad Up/Down

- Default: 125
- Min: 0
- Max: 250

Press the **D-Pad Up/D-Pad Down** button to increase/decrease the range of motion. The sensitivity may need to be increased in order to achieve the full range of motion.



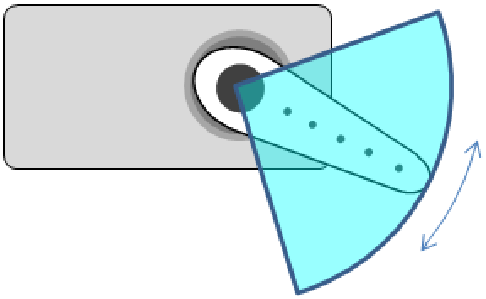
3.2.6 Range of Motion Offset Trim

D-Pad Left/Right

- Default: 0
- Min: -200

- Max: 200

Press the **D-Pad Left/Right** button to shift the servo's position offset, along with the range of motion's limits, left and right. Use the "Trim" option to adjust the servo's position without shifting the range of motion.

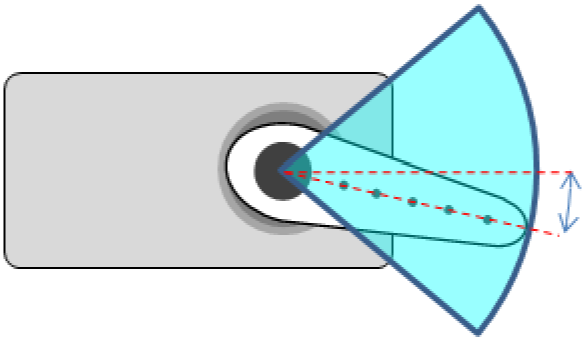


3.2.7 Center Offset Trim

R-Stick + D-Pad Left/D-Pad Right

- Default: 0
- Min: -200
- Max: 200

Servo Mode: Hold **R-Stick** and press **D-Pad Left** or **D-Pad Right** in order to adjust the servo's position offset without changing the range of motion's limits. Use this feature to compensate for any asymmetry in the servo's range of motion.



3.2.8 Home Position Calibration

PS Button / Hold PS Button (1 second)

The neutral position of the controller analog inputs can be calibrated by pressing the **PS Button**. Holding the **PS Button** for over 1 second will set the home position all sticks, triggers, tilt.

3.2.9 Servo Hold Position/Home Position Recall

L-Stick+Circle

Hold **L-Stick** and press **Circle** to cycle through the modes:

- OFF (default)
- Bind to L-Stick Button
- Bind to R-Stick Button

Hold **L-Stick** and press **Circle** to enable the hold position/home position function for a servo channel. When this is enabled and the channel is in position mode (see 3.2.1), tapping the joystick button it is tied to will freeze the value of that input channel until it is pressed again. When the channel is in incremental position mode, pressing the joystick button will return the servo to its home position. To set a new home position, hold the **PS Button** and press the **R-Stick/L-Stick Button**. This will set new home positions for all channels bound to the particular joystick button.

3.2.10 Trigger Link

L-Stick+Cross (valid when configuring a trigger channel)

- OFF (Default)
- Left/Right triggers linked

Hold **L-Stick** and press **Cross** to enable the trigger link mode. In this mode, the control input to the two servos tied to the trigger is a function of the difference between the trigger positions. This allows you to have the servo move in the opposite direction when the opposite trigger is pulled.

3.2.11 Touchpad Mode

R-Stick + D-Pad Up (valid when configuring a touchpad servo channel)

- Touchpad mode (default)
- Virtual joystick mode

In regular touchpad mode, the servo position is mapped to a specific point on the touchpad, and the servo will hold the position even after the finger has been lifted. When in this mode, setting the servo to incremental mode has no effect. In virtual joystick mode, the servo will move when you place your finger on the touchpad and slide it in a direction, and the input command is zeroed when the finger is lifted. Each servo channel is configured individually.

3.2.12 Touchpad Split

R-Stick + D-Pad Down (valid when configuring a touchpad servo channel)

- Touchpad split mode (default)
- Touchpad not split

The touchpad is in split mode, the servo channel will only move when the finger is initially touched down on that half of the touchpad.

Channel 6: Left half X-axis
Channel 7: Left half Y-axis
Channel 8: Right half x-axis
Channel 9: Right half y-axis

When the touchpad is not split, then that channel will accept an input no matter which side the finger is on. Each servo channel is configured individually.

3.2.13 Joystick Differential Drive Mixer

L-Stick+Cross (valid when configuring a joystick channel)

- Default: OFF

Hold the **L-Stick** and press **Cross** to enable the differential drive mixer for the joystick that you are currently configuring. The differential drive mixer changes the outputs of the x- and y-axis channels of the joystick to a differential drive control. Each axis channel can still be individually adjusted for sensitivity, range, direction, etc.

Channel 0/Channel 2: Left Wheel

Channel 1/Channel 3: Right Wheel

Wheel spin directions may need to be inverted, depending on your setup.

The Mecanum drive mixer, when enabled, will override the differential drive mixer.

3.2.14 Joystick Mecanum Drive Mixer

L-Stick+Square

The Mecanum Drive Mixer will mix channels 0-3 (the four joystick channels) to control four Mecanum wheels. This option, when enabled, will override the differential steering option. The controls and channel assignments are:

Left stick x-axis: Translate sideways
Left stick y-axis: Translate forwards/backwards
Right stick x-axis: rotate left/right
Right stick y-axis: unused, but can be remapped to a different output channel.

Channel 0: left front wheel

Channel 1: left rear wheel
Channel 2: right front wheel
Channel 3: right rear wheel

Wheel spin directions may need to be inverted, depending on your setup.

Press **L-Stick** + **Square** to cycle through the available modes:

- Mecanum Off (Default)
- Mecanum On, R-stick y-axis unused.
- Mecanum On, remap R-Stick y-axis to channel 4 output
- Mecanum On, remap R-Stick y-axis to channel 5 output
- Mecanum On, remap R-Stick y-axis to channel 6 output
- Mecanum On, remap R-Stick y-axis to channel 7 output
- Mecanum On, remap R-Stick y-axis to channel 8 output
- Mecanum On, remap R-Stick y-axis to channel 9 output
- Mecanum On, remap R-Stick y-axis to channel 10 output
- Mecanum On, remap R-Stick y-axis to channel 11 output

After remapping the channel, the configuration options in the remapped channel will apply to the joystick y-axis, *except* for the home position calibration (3.2.8); the calibration value for the original channel 3 will still apply.

3.3 Button Adjustments

3.3.1 Test Output

R-Stick Button

Press **R-Stick** to test the output of the selected channel when in configuration mode.
Since the buttons have already been mapped to various configuration commands, the right stick button is used to test each of the digital output behavior instead of the actual button mapped to the selected channel.

3.3.2 Invert Output

Cross

Press the **Cross** button to invert the output.

3.3.3 Button Output Modes

Triangle

- Default Value: Pushbutton

Press **Triangle** to cycle through different output modes. The available modes are:

- Pushbutton (output active when button is held)
- Toggle (output toggles when button is pressed)
- Single-Shot (output is active for a fixed duration when button is pressed)
- Autofire (output toggles on and off repeatedly when button is held)
- Toggle Autofire (same as the Autofire mode, but the output will continue to toggle until the button is pressed again)

3.3.4 Adjust Single-Shot/Autofire Pulse Width

D-Pad Left/Right

- Default: 50ms
- Min: 10ms
- Max: 600s
- Increments: 10ms
- Jitter: < 0.2ms

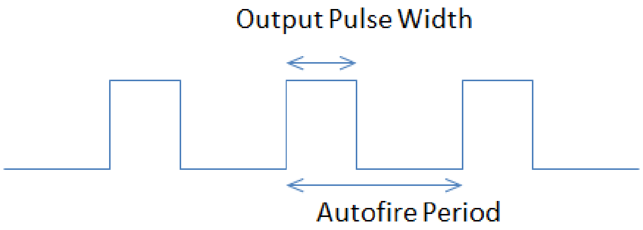
Press **D-Pad Left/Right** to adjust the output pulse width for single-shot/autofire modes. The output pulse will experience up to 0.2ms of jitter.

3.3.5 Adjust Autofire Period

D-Pad Up/Down

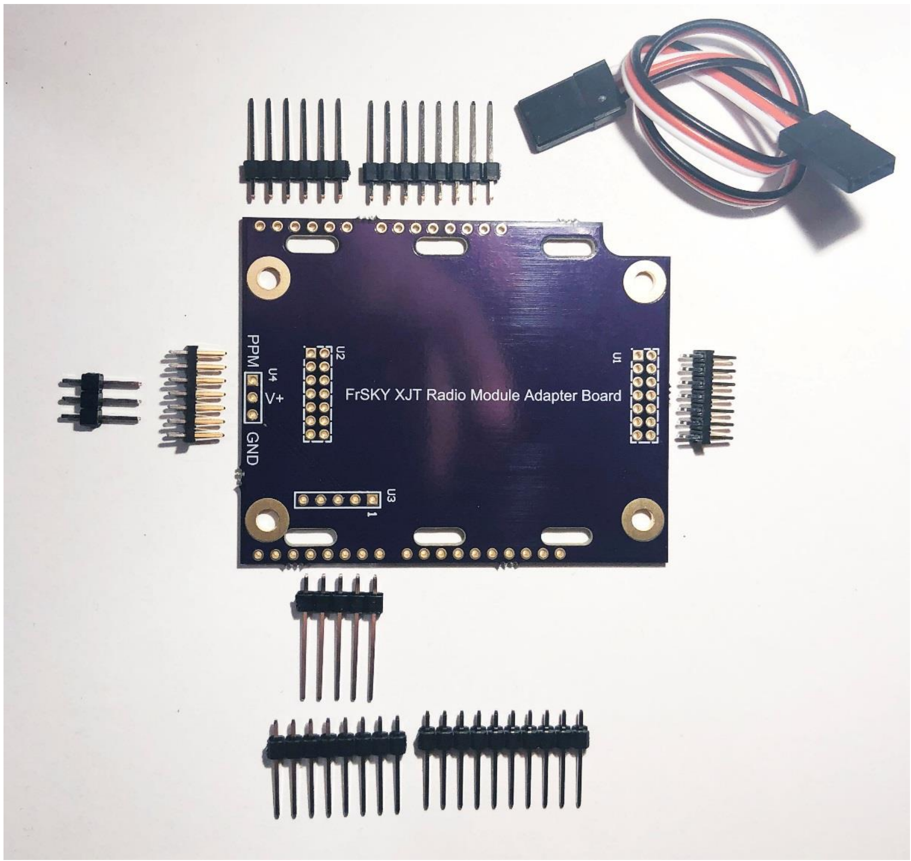
- Default: 100ms
- Min: 20ms
- Max: 600s
- Increments: 10ms
- Jitter: < 0.2ms

Press **D-Pad Up/Down** to adjust the autofire period. *The autofire period should be greater than the pulse width or the output will not toggle on and off.*

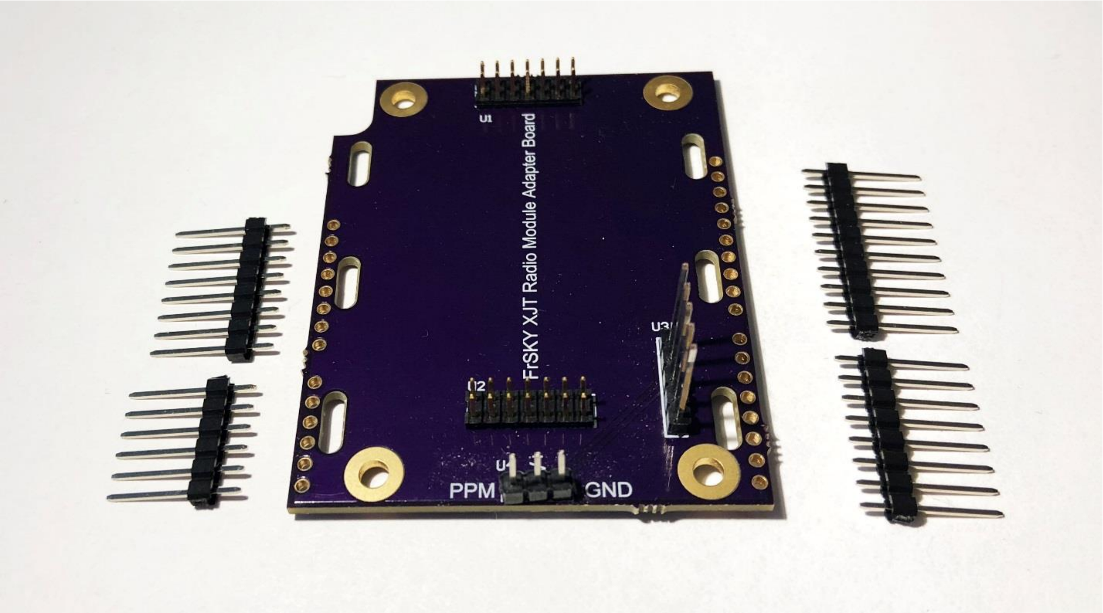


4.0 FrSKY XJT Radio Module Adapter

An adapter board kit is available for the FrSKY XJT transmitter module. Only the long 5-pin header U3 and the 3-pin header U4 have electrical connections. Header pin arrays U1 and U2 provide mechanical retention for the radio module, and the Arduino headers provide the option of stacking the board on top of the ServoShock. No electrical connections go through the Arduino headers. The slotted holes in the board allow you to fasten the radio module down with zip ties if desired.



The adapter board should look like this after assembly (the optional Arduino headers are not installed in this example).



To connect the ServoShock to the XJT radio, first enable the PPM mode option (see Section 3.1.10 PPM Output Mode for instructions). Use the 3-wire servo jumper cable to connect the U4 header on the adapter board to channel 9 on the ServoShock.

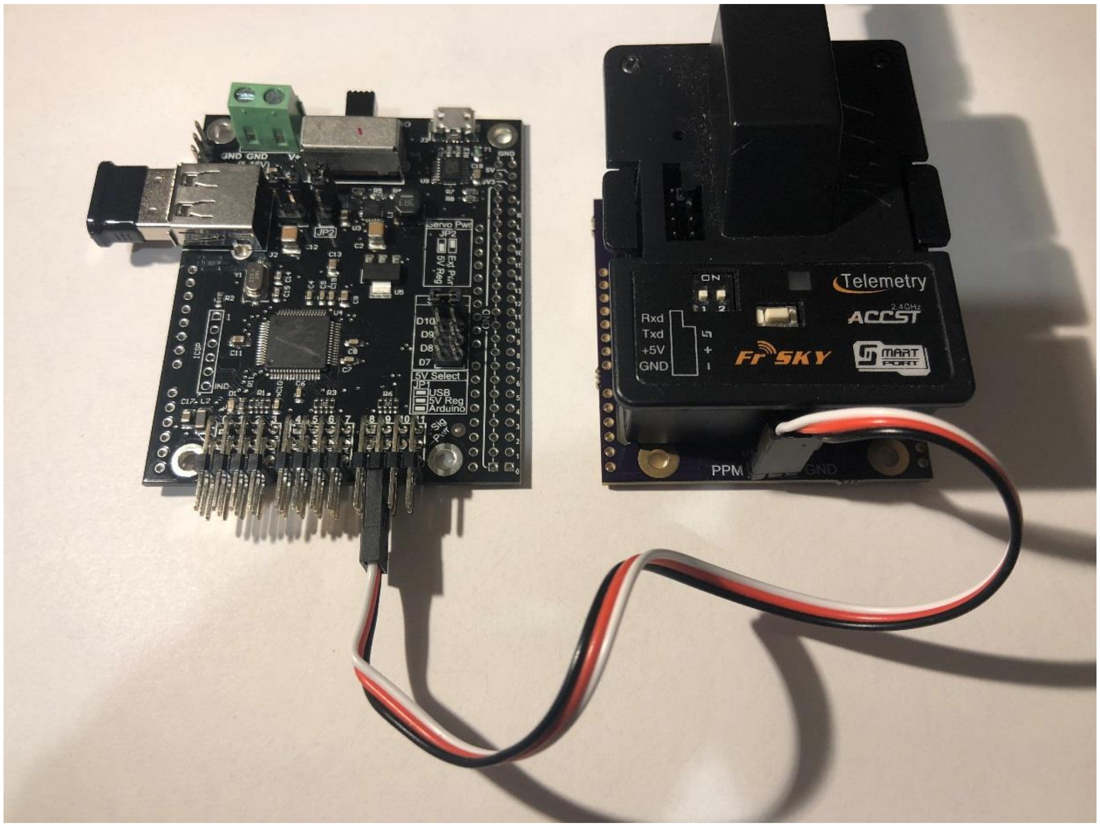
The FrSKY XJT transmitter has an input voltage spec of 6-15V, so we need to bypass the 5V regulator. Set jumper JP2 to the "Ext Pwr" setting (top position), and power the servoshock with 6-15V. **Do not plug servos into the ServoShock if your input voltage is over 6V, you will fry them.**

This setup was tested using the FrSKY X8R receiver (outputs servo channels 0-7) and the 4-port S.BUS-to-PWM decoder (outputs servo channels 8-11). The S.BUS-to-PWM decoder's PWM pulse train comes ever 9ms instead of the usual 20, so use digital servos instead. In the tested setup, the following were used:

- FrSKY XJT transmitter module
  - https://alofthobbies.com/frsky-xjt-jr-graupner-type-16ch-duplex-transmitter-telemetry-module.html
- FrSKY X8R receiver
  - https://alofthobbies.com/frsky-x8r-8-16-channel-receiver.html
- FrSKY S.BUS to PWM Decoder
  - https://alofthobbies.com/frsky-sbus-cppm-decoder-with-pins.html
- FrSKY Servo Channel Changer
  - https://alofthobbies.com/frsky-servo-channel-changer-sbus-cppm.html

- 8x analog servos (plugged into the X8R reciver)
- 4x digital servos (plugged into the S.BUS-to-PWM decoder)

Pair the XJT transmitter with your receiver following the instructions from FrSky, using mode 2 or mode 5 on the X8R receiver, and the ServoShock should start transmitting 12 RC servo channels to the receiver. Servo channels 0-7 on the ServoShock map to channels 1-8 on the X8R, and channels 8-11 on the ServoShock are output as channels 9-12 on the S.BUS output from the X8R. To get 9-12 from the receiver, use the S.BUS channel changer to program the S.BUS-to-PWM decoder to output the desired channels, and then plug the decoder into the S.BUS output port. The decoder's output signal has a faster pulse train (9ms period instead of 20ms); some analog servos may not tolerate this. Digital servos are recommended.



5.0 Electrical Specifications

Power Input Voltage: 5V-17V

Maximum Input Pin Voltage

Pins 10-13 (Rumble and LED): 3.3 V

Pins 15, 21-24 (UART Rx, SPI BUS and Slave Select): 5.5 V

Current Consumption (nominal)

6V Input:

Controller Disconnected: 3 mA

Controller Connected: 18 mA

12V Input:

Controller Disconnected: 2 mA

Controller Connected: 12 mA

Maximum source/sink current for any I/O pin: 25 mA

Maximum combined source/sink current for all I/O pins: 200 mA

UART Baud Rate: 115200 bps

Maximum SPI Bus Clock Frequency: 1 MHz

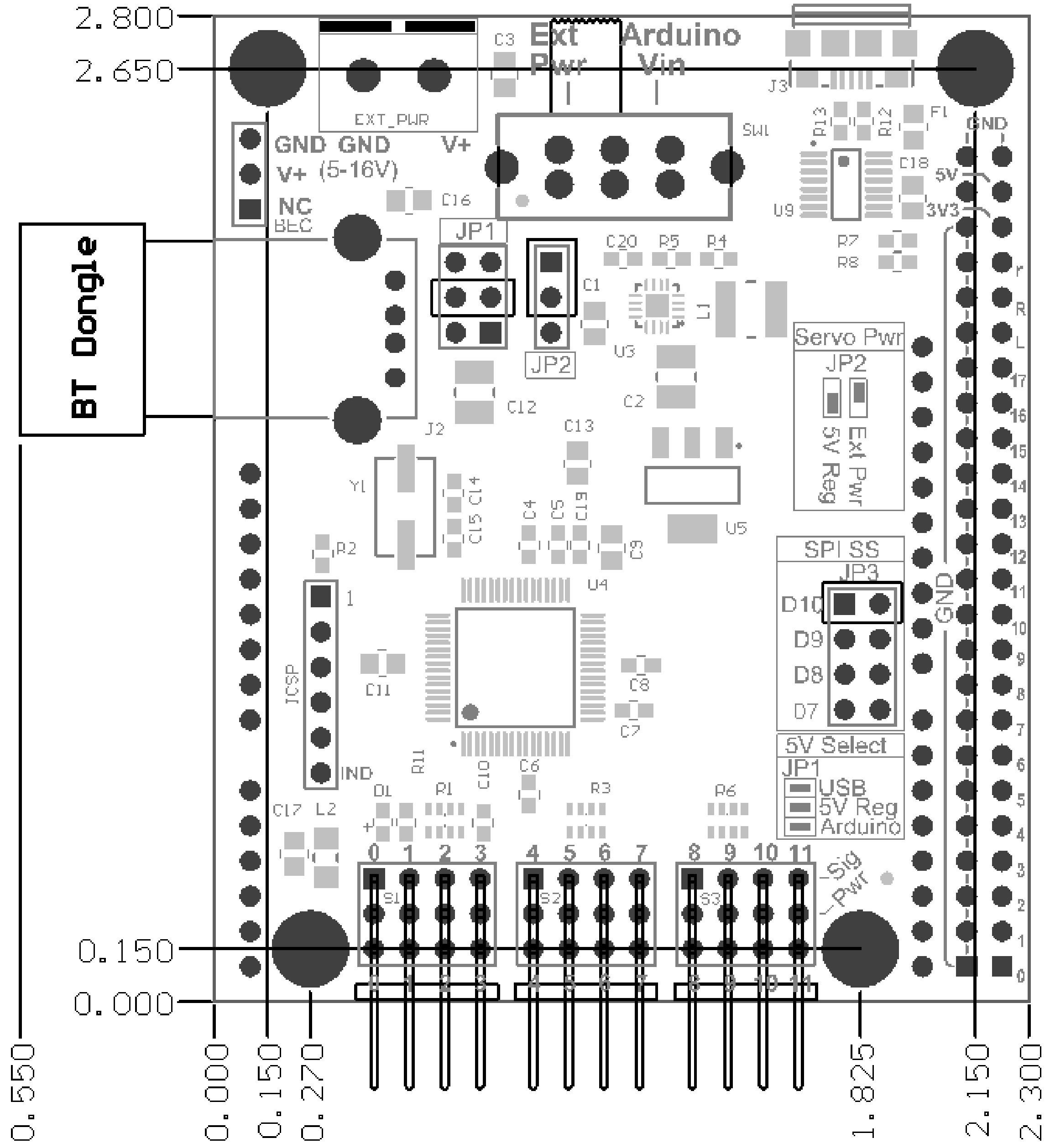
PS4 Controller range: 10 Meters

Controller input poll rate: 100 Hz

Controller LEDs and rumble update rate: 10 Hz

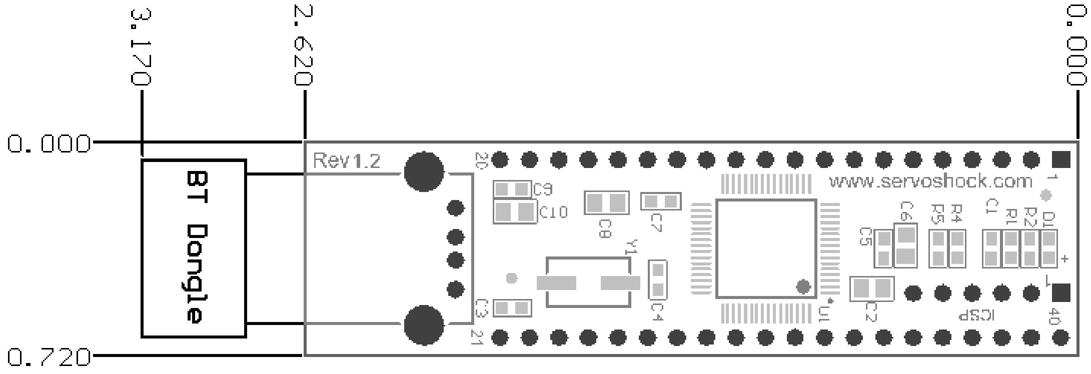
6.0 PHYSICAL DIMENSIONS

6.1 ServoShock Board



*Dimensions are in inches.

6.2 DIP-40 Module



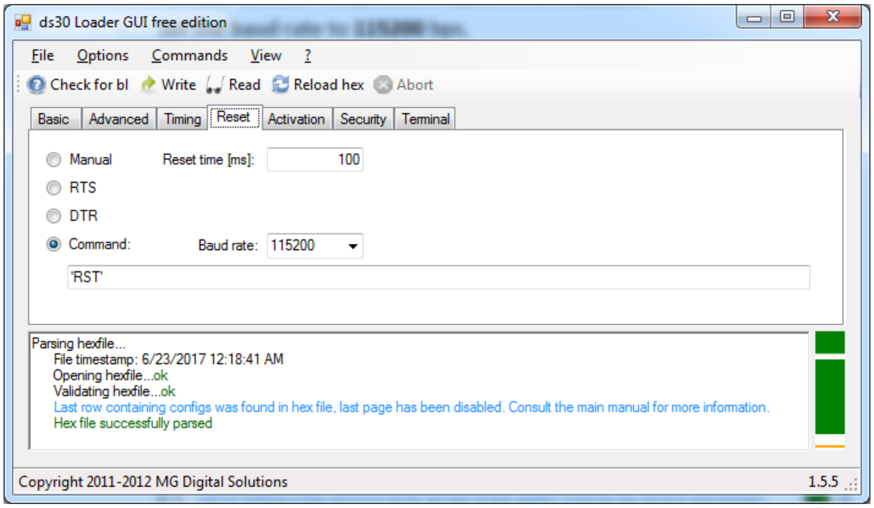
*Dimensions are in inches.

7.0 REPROGRAMMING FIRMWARE

New firmware can be loaded onto the microcontroller via the UART, using the DS30 bootloader (http://www.servoshock.com/documentation.html) for PIC24s. The bootloader configured for the ServoShock can be obtained at www.servoshock.com.

To reprogram the firmware, use the **ds30LoaderGUI.exe** file in the ds30 bootloader's **ds30 Loader\bin** folder.

1. Download the ServoShock source code .zip file and extract the files. Load the hex file in location **PIC_PS4_Host.X\dist\default\production\PIC_PS4_Host.X.production.hex**
2. For the "Device" field, select **PIC24FJ** and **64GB106**.
3. Set the baud rate to **115200** bps.
4. Set the port to the serial port the ServoShock is connected to.
5. Select the **Reset** tab and configure the fields as shown:

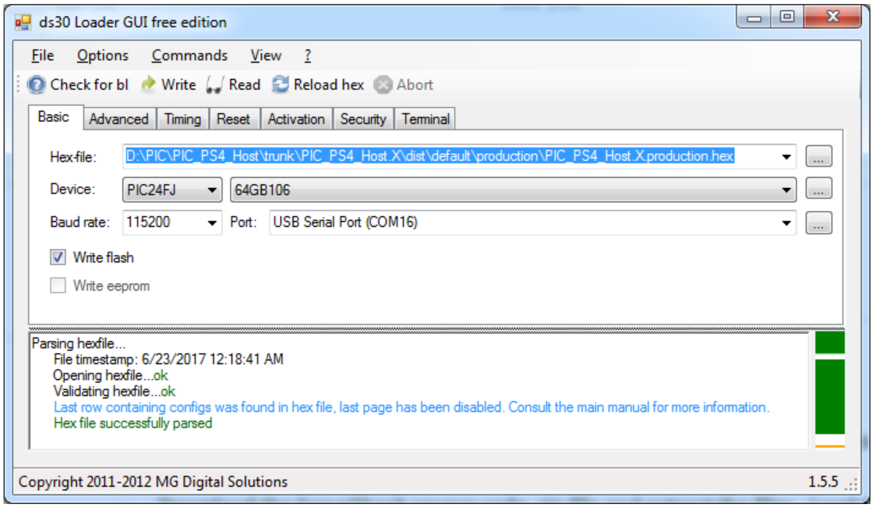

ds30 Loader GUI free edition
File Options Commands View ?
Check for bl Write Read Reload hex Abort
Basic Advanced Timing Reset Activation Security Terminal

○ Manual Reset time [ms]: 100
○ RTS
○ DTR
● Command: Baud rate: 115200
'RST'

```
Parsing hexfile...
    File timestamp: 6/23/2017 12:18:41 AM
    Opening hexfile...ok
    Validating hexfile...ok
    Last row containing configs was found in hex file, last page has been disabled. Consult the main manual for more information.
    Hex file successfully parsed
```

Copyright 2011-2012 MG Digital Solutions 1.5.5

6. On the **Basic** tab, check the **Write flash** box, and hit **Write.**


ds30 Loader GUI free edition
File Options Commands View ?
Check for bl Write Read Reload hex Abort
Basic Advanced Timing Reset Activation Security Terminal

Hex-file: D:\PIC\PIC_PS4_Host\trunk\PIC_PS4_Host.X\dist\default\production\PIC_PS4_Host.X.production.hex
Device: PIC24FJ 64GB106
Baud rate: 115200 Port: USB Serial Port (COM16)
☑ Write flash
□ Write eeprom

```
Parsing hexfile...
    File timestamp: 6/23/2017 12:18:41 AM
    Opening hexfile...ok
    Validating hexfile...ok
    Last row containing configs was found in hex file, last page has been disabled. Consult the main manual for more information.
    Hex file successfully parsed
```

Copyright 2011-2012 MG Digital Solutions 1.5.5

7. **(Important!)** After uploading new firmware, enter the configuration menu and hold down the Share button to reset all the setting to default (Section 3.1.3). This is required to reset the non-volatile memory addresses that hold device settings.

8.0 USING THE PS3 OR MOVE NAVIGATOR CONTROLLER

If you wish to use the PS3 controller or the PS3 Move Navigator, the ServoShock can be reprogrammed with the ServoShock 1 firmware. The firmware and documentation for the ServoShock 1 can be found at **www.servoshock.com/documentation**.

9.0 ARDUINO LIBRARY

The ServoShock Arduino library lets you interface the ServoShock with an Arduino Uno over the SPI bus. Jumper JP3 on the ServoShock sets the slave select pin to D7, D8, D9, or D10. The software library and an example Arduino program can be downloaded at **www.servoshock.com/documentation**.

9.1 Program Structure

The basic structure of the program looks like this:

```
#include <SPI.h>
#include "servoshock_PS4.h"

// set the slave select pin for the ServoShock.
// Set jumper JP3 on the Shield to D10 if using digital output 10.
const int slaveSelect = 10;
//create instance of ServoShock
ServoShock ServoShock1(slaveSelect);

void setup()
{
//initialize SPI:
    digitalWrite(slaveSelect, HIGH);
    SPI.setDataMode(SPI_MODE0);
    SPI.setClockDivider(SPI_CLOCK_DIV16);
    SPI.setBitOrder(MSBFIRST);
    SPI.begin();
}

void loop()
{
    //This updates the inPacket and outPacket structures.
    ServoShock1.Update();

    //Process inputs,  for example:
    if (ServoShock1.inPacket.dPadUp){...}

    //Process outputs. For example, if you want to control a servo:
    //enable servo override
    ServoShock1.outPacket.overrideLStickX = 1;
    //output this pulse width in microseconds.
    ServoShock1.outPacket.lStickX_uS = servo_uS;

    //Necessary if your program is short.  Don't update faster than 100Hz.
    delay(10);
}
```

When the `ServoShock::Update` function is run, data packets between the ServoShock and the Arduino are exchanged. Data from the ServoShock is stored in the `inPacket` struct, and data stored in the `outPacket` struct is sent to the ServoShock. Therefore, the effect from setting the bits won't take place until the `Update` function is called. Leave at least 10ms between calls of `Update` to prevent the ServoShock from being overloaded with SPI messages.

9.2 Reading Inputs

The state of the controller and the servo and digital output pins can be determined by reading the members of the `inPacket` structure.

`inPacket` Struct Members		
Structure Member	Data Type	Description
Dualshock 4 Controller State		
`lStickX`	uint8	Left stick X-axis
`lStickY`	uint8	Left stick Y-axis
`rStickX`	uint8	Right stick X-axis
`rStickY`	uint8	Right stick Y-axis
`lTriggerAnalog`	uint8	Left trigger analog
`rTriggerAnalog`	uint8	Right triger analog
`dPadUp`	1-bit	Direction pad up
`dPadRight`	1-bit	Direction pad right
`dPadDown`	1-bit	Direction pad down
`dPadLeft`	1-bit	Direction pad left
`lBumper`	1-bit	Left bumper
`rBumper`	1-bit	Right bumper
`square`	1-bit	Square button
`cross`	1-bit	Cross button
`circle`	1-bit	Circle button
`triangle`	1-bit	Triangle button
`lTriggerDigital`	1-bit	Left trigger digital
`rTriggerDigital`	1-bit	Right trigger digital
`share`	1-bit	Share button
`options`	1-bit	Options button
`lStickPress`	1-bit	Left stick button
`rStickPress`	1-bit	Right stick button
`psButton`	1-bit	PS button
`tPadPress`	1-bit	Touch pad button
`gyroX`	int16	X-axis gyroscope
`gyroY`	int16	Y-axis gyroscope
`gyroZ`	int16	Z-axis gyroscope
`accelX`	int16	X-axis accelerometer
`accelY`	int16	Y-axis accelerometer
`accelZ`	int16	Z-axis accelerometer
`tpad[0].finger[0].touchID`	uint8	ID assigned to the touch event of first finger
`tpad[0].finger[0].noFinger`	uint8	0 if finger is detected, 1 if no finger found
`tpad[0].finger[0].x`	uint16	x-position of first finger touch
`tpad[0].finger[0].y`	uint16	y-position of first finger touch

tpad[0].finger[1].touchID	uint8	ID assigned to the touch event of second finger
tpad[0].finger[1].noFinger	uint8	0 if finger is detected, 1 if no finger found
tpad[0].finger[1].x	uint16	x-position of second finger touch
tpad[0].finger[1].y	uint16	y-position of second finger touch
battery	4-bits	Battery level (>=7 is fully charged, 0 is empty)
ServoShock Pin Output States		
lStickXState_uS	uint16	Left stick x-axis servo signal pulse width in µS
lStickYState_uS	uint16	Left stick y-axis servo signal pulse width in µS
rStickXState_uS	uint16	Right stick x-axis servo signal pulse width in µS
rStickYState_uS	uint16	Right stick y-axis servo signal pulse width in µS
lTriggerAnalogState_uS	uint16	Left trigger servo signal pulse width in µS
rTriggerAnalogState_uS	uint16	Right trigger servo signal pulse width in µS
lTpadXState_uS	uint16	Left-side touchpad x-position servo signal pulse width in µS
lTpadYState_uS	uint16	Left-side touchpad y-position servo signal pulse width in µS
rTpadXState_uS	uint16	Right-side touchpad x-position servo signal pulse width in µS
rTpadYState_uS	uint16	Right-side touchpad y-position servo signal pulse width in µS
tiltXState_uS	uint16	Tilt x-axis servo signal pulse width in µS
tiltYState_uS	uint16	Tilt y-axis servo signal pulse width in µS
dPadUpState	1-bit	Direction pad up output pin state
dPadRightState	1-bit	Direction pad right output pin state
dPadDownState	1-bit	Direction pad down output pin state
dPadLeftState	1-bit	Direction pad left output pin state
triangleState	1-bit	Triangle button output pin state
circleState	1-bit	Circle button output pin state
crossState	1-bit	Cross button output pin state
squareState	1-bit	Square button output pin state
lBumperState	1-bit	Left bumper output pin state
rBumperState	1-bit	Right bumper output pin state
lTriggerDigitalState	1-bit	Left trigger output pin state
rTriggerDigitalState	1-bit	Right trigger output pin state
lStickPressState	1-bit	Left stick button output pin state
rStickPressState	1-bit	Right stick button output pin state
shareState	1-bit	Share button output pin state
optionsState	1-bit	Options button output pin state
tpadPressState	1-bit	Touchpad button output pin state
psButtonState	1-bit	PS button output pin state

9.3 Writing Outputs

The Arduino can control the ServoShock outputs as well as the rumble motors and the LED lightbar by writing to the `outPacket` structure. To control an output of the ServoShock, set the override bit of the output channel you wish to control. This will cause the ServoShock to ignore the input from the Dualshock 4 controller and instead use the value read in the `outPacket` structure. Clearing the override bit will revert control back to the Dualshock controller.

`outPacket` Struct Members		
Structure Member	Data Type	Description
	Override control bits	
`overrideLED`	1-bit	LED lightbar override
`overrideRumbleL`	1-bit	Low frequency rumble motor override
`overrideRumbleH`	1-bit	High frequency rumble motor override
`overrideLStickX`	1-bit	Left stick x-axis servo override
`overrideLStickY`	1-bit	Left stick y-axis servo override
`overrideRStickX`	1-bit	Right stick x-axis servo override
`overrideRStickY`	1-bit	Right stick y-axis servo override
`overrideLTriggerAnalog`	1-bit	Left trigger servo override
`overrideRTriggerAnalog`	1-bit	Right trigger servo override
`overrideLTpadX`	1-bit	Left-side touchpad servo x-axis override
`overrideLTpadY`	1-bit	Left-side touchpad servo y-axis override
`overrideRTpadX`	1-bit	Right-side touchpad servo x-axis override
`overrideRTpadY`	1-bit	Right-side touchpad servo y-axis override
`overrideTiltX`	1-bit	Tilt x-axis servo override
`overrideTiltY`	1-bit	Tilt y-axis servo override
`overrideDPadUp`	1-bit	Direction pad up override
`overrideDPadRight`	1-bit	Direction pad right override
`overrideDPadDown`	1-bit	Direction pad down override
`overrideDPadLeft`	1-bit	Direction pad left override
`overrideTriangle`	1-bit	Triangle button override
`overrideCircle`	1-bit	Circle button override
`overrideCross`	1-bit	Cross button override
`overrideSquare`	1-bit	Square button override
`overrideLBumper`	1-bit	Left bumper override
`overrideRBumper`	1-bit	Right bumper override
`overrideLTriggerDigital`	1-bit	Left trigger digital output override
`overrideRTriggerDigital`	1-bit	Right trigger digital output override
`overrideLStickPress`	1-bit	Left stick button override
`overrideRStickPress`	1-bit	Right stick button override

overrideShare	1-bit	Share button override
overrideOptions	1-bit	Options button override
overrideTpadPress	1-bit	Touchpad button override
overridePsButton	1-bit	PS button override
Output State Registers		
lStickX_uS	uint16	Left stick x-axis servo signal pulse width in µS
lStickY_uS	uint16	Left stick y-axis servo signal pulse width in µS
rStickX_uS	uint16	Right stick x-axis servo signal pulse width in µS
rStickY_uS	uint16	Right stick y-axis servo signal pulse width in µS
lTriggerAnalog_uS	uint16	Left trigger servo signal pulse width in µS
rTriggerAnalog_uS	uint16	Right trigger servo signal pulse width in µS
lTpadX_uS	uint16	Left-side touchpad x-position servo signal pulse width in µS
lTpadY_uS	uint16	Left-side touchpad y-position servo signal pulse width in µS
rTpadX_uS	uint16	Right-side touchpad x-position servo signal pulse width in µS
rTpadY_uS	uint16	Right-side touchpad y-position servo signal pulse width in µS
tiltX_uS	uint16	Tilt x-axis servo signal pulse width in µS
tiltY_uS	uint16	Tilt y-axis servo signal pulse width in µS
dPadUp	1-bit	Direction pad up output pin state
dPadRight	1-bit	Direction pad right output pin state
dPadDown	1-bit	Direction pad down output pin state
dPadLeft	1-bit	Direction pad left output pin state
triangle	1-bit	Triangle button output pin state
circle	1-bit	Circle button output pin state
cross	1-bit	Cross button output pin state
square	1-bit	Square button output pin state
lBumper	1-bit	Left bumper output pin state
rBumper	1-bit	Right bumper output pin state
lTriggerDigital	1-bit	Left trigger digital output pin state
rTriggerDigital	1-bit	Right trigger digital output pin state
lStickPress	1-bit	Left stick button output pin state
rStickPress	1-bit	Right stick button output pin state
share	1-bit	Share button output pin state
options	1-bit	Options button output pin state
tpadPress	1-bit	Touchpad button output pin state
psButton	1-bit	PS button output pin state

9.3.1 Setting LEDs

When setting the color of the LEDs, the values can be written directly to the `outPacket` struct, or the `SetLED` function can be used.

Function:

```
void ServoShock::SetLED(unsigned char red, unsigned char green,
                        unsigned char blue, unsigned char blinkOnDuration,
                        unsigned char blinkOffDuration)
```

Parameters:

- red: Red, values from 0-255.
- green: Green, values from 0-255.
- blue: Blue, values from 0-255.
- blinkOnDuration: This enables the blinking mode and sets the LED on time. Values are from 0-255, each count corresponds to about 10ms.
- blinkOffDuration: This enables the blinking mode and sets the LED off time. Values are from 0-255, each count corresponds to about 10ms.

10.0 REVISION HISTORY

Revision	Date	Notes
1.4	5/1/18	Initial Release
1.5	5/14/18	PPM mode is on CH9, not 10
1.6	5/17/18	Added notes on using the XJT / X8R radio