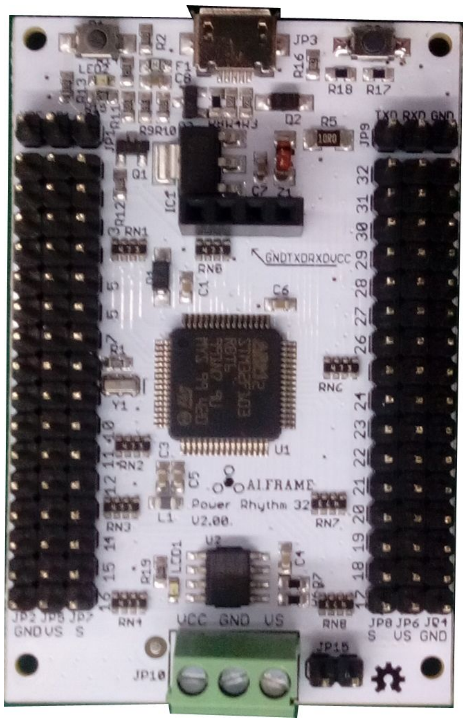
AI.FRAME

Power Rhythm 32 Guide V1.0



Cautions!

- Read this quick start guide completely before wiring and applying power to the board! Errors in wiring can damage the Power Rhythm 32 board, STM32 or EEPROM Chip, and any attached servos or peripherals.
- Never reverse the power coming in to the board. Make sure the black wire goes to (-) GND, and the red wire goes to (+) VCC, or VS. Never connect peripherals when the board is powered on.
- The on-board regulator can provide 250mA total. This includes the micro controller chip, the on-board LEDs, and any attached peripherals. Drawing too much current can cause the regulator to overheat.

Power Rhythm 32

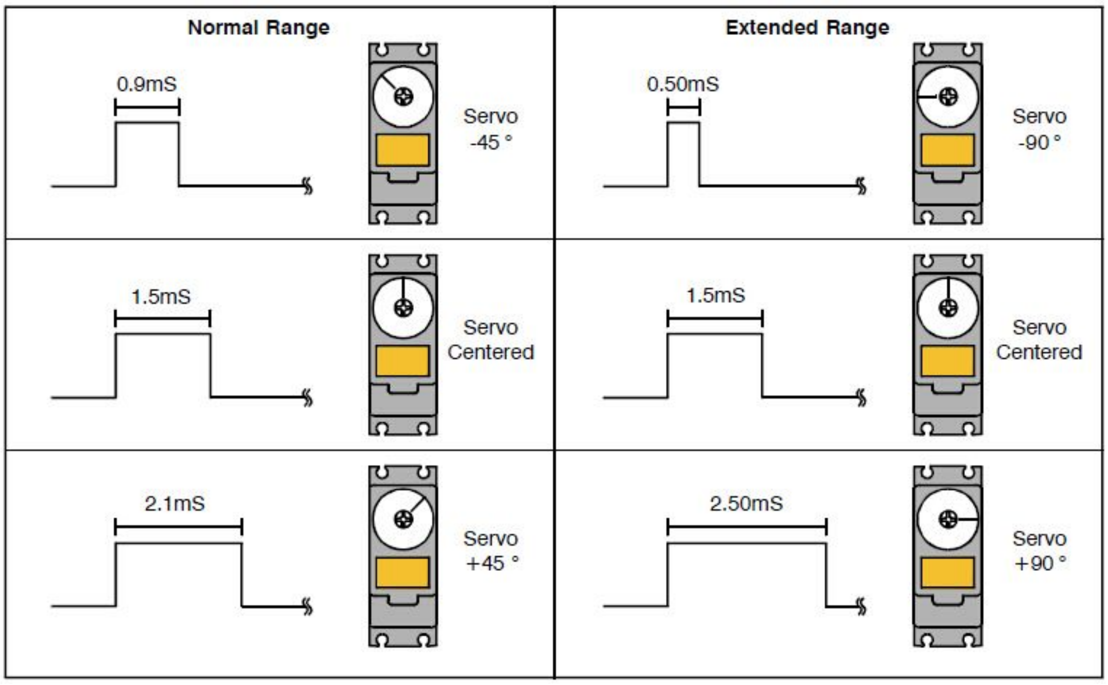
The Power Rhythm 32 is a servo controller allows you to control up to 32 servos by sending serial commands from a micro controller or PC via TTL serial or USB connection.Since the Propeller The Power Rhythm 32 is powered by a STM32 chip and base on an Arduino compatible board Maple Rev 3, the firmware is open-source, so the Power Rhythm 32 is also a development platform which can be customized for specific applications including systems that require servos, DC motors, stepper motors and lighting control.

Maple quick start guide:

http://leaflabs.com/docs/hardware/maple.html

What is a Servo?

Before we illustrate how to use the servo controller we need to explain what a servo is, and define the control methodology.
Pulse-proportional servos are designed for use in radio-controlled (R/C) cars, boats and planes. They provide precise control for steering, throttle, rudder, etc. using a signal that is easy to transmit and receive. The signal consists of positive going pulses ranging from 0.9 to 2.1mS (milliseconds) long, repeated 50 times a second (every 20mS). The servo positions its output shaft in proportion to the width of the pulse, as shown below.
In radio-control applications, a servo needs no more than a 90° range of motion, since it is usually driving a crank mechanism that can't move more than 90°. So when you send pulses within the manufacturer-specified range of 0.9 to 2.1mS, you get around 90° range of motion.
Most servos have more than 90° of mechanical range. In fact, most servos can move up to 180° of rotation. However, some servos can be damaged when commanded past their mechanical limitations. The SSC-32 lets you use this extra range. A position value of 500 corresponds to 0.50mS pulse, and a position value of 2500 corresponds to a 2.50mS pulse. A one unit change in position value produces a 1uS (microsecond) change in pulse width. The positioning resolution is 0.09°/unit (180°/2000). From here on, the term pulse width and position are the same.



Remember that some servos may not be able to move the entire 180° range. Use care when testing servos. Move to the extreme left or right slowly, looking for a point when additional positioning values no longer result in additional servo output shaft movement. When this value is found, put it as a limit in your program to prevent damaging the servo. Generally, micro servos are not able to move the entire 180° range.

Feathers:

- STM32 based hardware
- Program via USB interface with PC
- Separate screw-terminal power supply for servos and chips
- Servo speed controlled and acceleration controlled
- Compatible with Arduino coding
- The accuracy of PWM signal is 1 uS
- Support all kinds of PWM servos
- Open-source firmware and hardware
- Support offline mode
- 512K EEPROM flash for offline command data saving

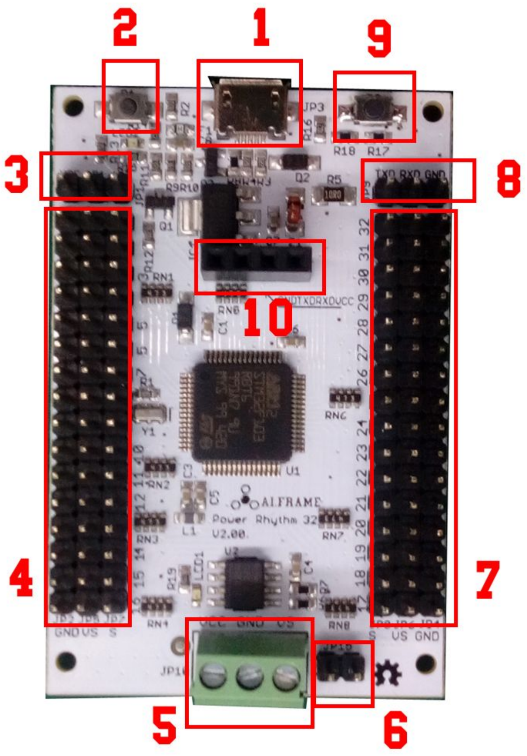
Application:

- Fun-house prop control
- Animated art control
- Robotics

Specifications：

- Power requirements: 5 - 12 VDC @ ～60 mA for logic VCC, 4.8 – 7.5 VDC for servos VS (do not exceed your servos’input voltage rating)
- Communication: Asynchronous Serial @ 115200 bps (TTL or USB)
- Operating temperature: 32 to 158 °F (0 to 70 °C)
- Dimensions: 2.40 x 1.60 x 0.45 in (61.0 x 40.6 x 11.5 mm)

Overview:



1. Micro USB port
2. Reset button P1
3. Analog input AD0, AD1, and 3.3V DC output VCC
4. Servo signals output 1 - 16. A servo has three wires: (-), (+), and signal, (-) to row GND, (+) to row VS, and signal to row S. The color of signal wire usually is orange or white.
5. Power supply for chips (VCC) and servos (VS)
6. JP15: connect these two pins will connect VCC to VS
7. Servo signals output 17 - 32.A servo has three wires: -, +, and signal, - to row GND, + to row VS, and signal to S. The color of signal wire usually is orange or white.
8. UART port
9. Button P2
10. Bluetooth module port

Install the Drivers:

- Download the ZIP file for 32-bits Windows (http://static.leaflabs.com/pub/leaflabs/maple-ide/maple-ide-0.0.12-windowsxp32.zip)
- Extract all the files in the ZIP file to a suitable location on your system (like your Desktop folder).
- First, install DFU drivers (for uploading code to your Power Rhythm 32) using the following steps.
  1.Plug your Power Rhythm 32 into the USB port.

2.Hit the reset button P1 on your Power Rhythm 32 (it's the small button at the top left, labeled P1). Notice that it blinks quickly 6 times, then blinks slowly a few more times.
3.Hit reset again, and this time push and hold the other button during the 6 fast blinks (the button is on the top right; it is labeled P2). You can release it once the slow blinks start.
4.Your Power Rhythm 32 is now in perpetual bootloader mode. This should give you a chance to install the DFU drivers.
5.Windows should now prompt you for some drivers. In the top level directory of the Maple IDE, point Windows to drivers/mapleDrv/dfu/.

- Next, install serial drivers (for communicating with your Power Rhythm 32 using serial over USB).
1.Reset your Power Rhythm 32 and allow it to exit the bootloader (wait for the slow blinking to stop). The Power Rhythm 32 will next start running whatever program was uploaded to it last. (New Maples will start running the test program we upload to them before shipping them to you).
2.Once Power Rhythm 32 is running some user code, Windows should prompt you for more drivers. Point windows to driver/mapleDrv/serial. You can now run the Servo_Rhythm_Controller.exe for 32-bits Windows.

Command Formatting:

- Servo Movement:
#<ch>P<pw>... #<ch>P<pw>T<time>!
# = Command start flag
<ch> = Channel number in decimal, 1 - 32.
<pw> = Pulse width in microseconds, 500 - 2500.
<time> = Time in mS for the entire move, affects all channels, 65535 max.
! = Command end flag.
It returns an "N", when this command finishes.
Servo Move Example: "#5P1600T1000!"
The example will move servo 5 to position 1600. It will take 1 second to complete the move regardless of how far the servo has to travel to reach the destination.
Multi Servo Move Example: "#5P1600#10P750T2500!"
The example will move servo 5 to position 1600 and servo 10 to position 750. It will take 2.5 seconds to complete the move, even if one servo has farther to travel than another. The servos will both start and stop moving at the same time. This is a very powerful command. By commanding all of the legs in a walking robot with the Group Move it is easy to synchronize complex gaits. The same synchronized motion can benefit the control of a robotic arm as well.
- Read Analog Inputs:
AD<ch>!
<ch> = Channel number in decimal, 0 - 1.

Analog read example: “AD1!”

Read the value on the pin AD1 as analog, it returns the decimal value from 0 to 4095, A return value of 0 represents 0vdc. A return value of 4095 represents +3.3vdc

- Offline Command Group:

  EN<ch>!

  <ch> = Channel number in decimal, 0 - 48.

  If <ch> is not 0, Run the offline servo move command saved in the flash, otherwise disable the offline servo group move command. It returns an “W”, when the Power Rhythm 32 gets this command.

  Offline Group Move Example: ”EN1!”

  Circle to run the offline servo move command in the group 1. One group can contain up to 20 servo group move commands.

- Circle All Offline Command Group: ”GO!”

  This command will circle to run all the command in the flash, start with group 1. It returns an “W”, when this command finishes.

- Read The Number of Offline Command Group:”READ!”

  It returns the number of the offline command group.

- Check The Version Of Firmware: ”V!”

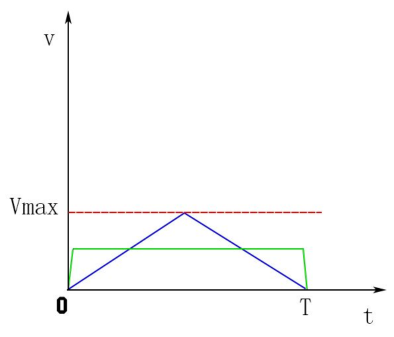
  It returns the version of the Power Rhythm 32.

- Change The Acceleration Mode

  SM<ch>!

“SM0!”: Without acceleration control, Speed change probably like the green line in the figure.

“SM1!”:Speed changing according to Trapezoidal_curve. Speed change probably like the blue line in the figure. This can decrease the shaking when a servo coma to a stop.



AI.FRAME

Hardware design: https://github.com/AiFrame/Power_Rhythm_32/tree/master/Eagle_files

Software design: https://github.com/AiFrame/Power_Rhythm_32/tree/master/Firmware

Website: http://aiframe.me

Forum: http://forum.aiframe.me

E-mail: support@aiframe.me