Parallax Continuous Rotation Servo (#900-00008)

The Parallax Standard Servo is ideal for robotics and basic movement projects.

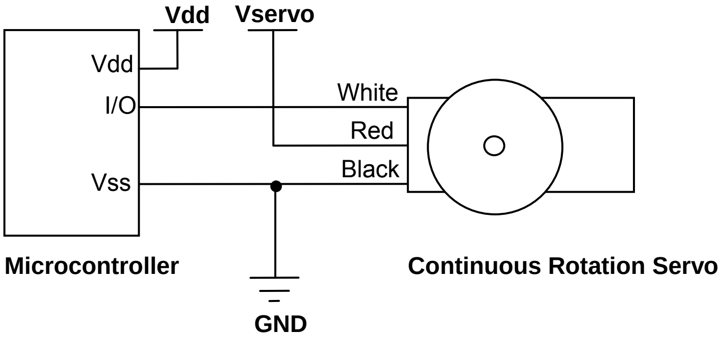
Features

- Bidirectional continuous rotation
- 0 to 50 RPM, with a linear response to PWM for easy ramping
- Accepts four mounting screws
- Easy to interface with any Parallax microcontroller or PWM-capable device
- Very easy to control with PBASIC's or SX/B's PULSOUT commands
- Manufactured for Parallax exclusively by Futaba

Technical Specifications

- Power requirements: 4 to 6 VDC
- Maximum current draw: 140 +/- 50 mA at 6 VCDC when operating in no load conditions
  15 mA when in static state
- Communication: pulse-width modulation; TTL/CMOS 3.3 to 5V
- Dimensions: approx 2.2 x 0.8 x 1.6 in (5.58x 1.9 x 40.6 cm) excluding servo horn
- Operating temperature range: 14 to 122°F (-10 to 50°C)
- Weight: 1.50 oz (42.5 g)

Quick-Start Circuit



Vdd = microcontroller voltage supply

Vservo = 4 to 6 VDC, regulated or battery (See Board of Education Servo Header Connection Diagram, page 2)

I/O = PWM TTL or CMOS output signal, 3.3 to 5 V, not to exceed Vservo + 0.2 V

Device Information

The Parallax continuous rotation servo relies on pulse width modulation to control the speed and direction of the servo shaft. Before utilizing the servo in a project, it is important to calibrate the center position of the servo in order to define the point where the servo is at rest (see Calibration – “Center” the Servo below).

Specifications

Pin	Name	Description	Minimum	Typical	Maximum	Units
1 (White)	Signal	Input; TTL or CMOS	3.3	5.0	Vservo + 0.2	V
2 (Red)	Vservo	Power Supply	4.0	5.0	6.0*	V
3 (Black)	Vss	Ground		0		V

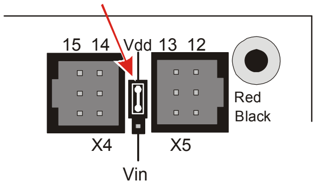
*See Board of Education Servo Header Connection Diagram, page 2.

Power Precautions

- Do not use this servo with an unregulated wall-mount supply. Such power supplies may deliver variable voltage far above the stated voltage.
- Do not power this servo through the BASIC Stamp Module's Vin pin, this can deliver voltages above the stated voltage. See the Board of Education Connection Diagram below for jumper settings.
- Servo current draw can spike while under peak load; be sure your application's regulator is prepared to supply adequate current for all servos used in combination.

Board of Education Servo Header Connection Diagram

When connecting the servo to the Board of Education Rev C or higher’s servo header, be sure the jumper is set to Vdd as shown in the figure below. Failure to place the jumper at this setting can cause damage your servo!



Calibration – "Center" the Servo

The servo has a potentiometer access port, allowing the user to adjust the servo to hold completely still when receiving a 1.5 ms pulse width. This is the value in the "center" of the range of control pulses the servo will accept.

To center the servo, program your host device to deliver a 1.5 ms pulse, continually refreshed every 20 ms. Sample calibration code is given below for all BASIC Stamp models, Spin for the Propeller, and SX/B for the SX chip. All are available for download from the 900-00008 product page at www.parallax.com.

Connect the servo to your microcontroller's I/O pin. The example programs below specify an I/O pin. Program

BASIC Stamp Calibration Code - for all BS2 models

- √ Connect the servo to BASIC Stamp 1/O pin P12, or update the ToServo PIN declaration.
- √ Run the program, and gently twist the potentiometer adjustment screw until the servo does not turn or vibrate. *NOTE: Calibrating the servo may take some patience. The potentiometer is very sensitive so a very light touch will be required.*

```
' {$STAMP BS2}
' {$PBASIC 2.5}

#SELECT $Stamp
  #CASE BS2, BS2E, BS2PE         ' PULSOUT Duration units are 2 us for these models
    Center CON 750
  #CASE BS2SX, BS2P, BS2PX       ' PULSOUT Duration units are 0.8 us for these models
    Center CON 1875
#ENDSELECT

ToServo PIN 12                   ' connect servo to I/O pin P12, or change it here

DO
  PULSOUT ToServo, Center        ' ToServo pin outputs 1.5 ms pulse
  PAUSE 20                       ' refresh pulse every 20 milliseconds
LOOP
```

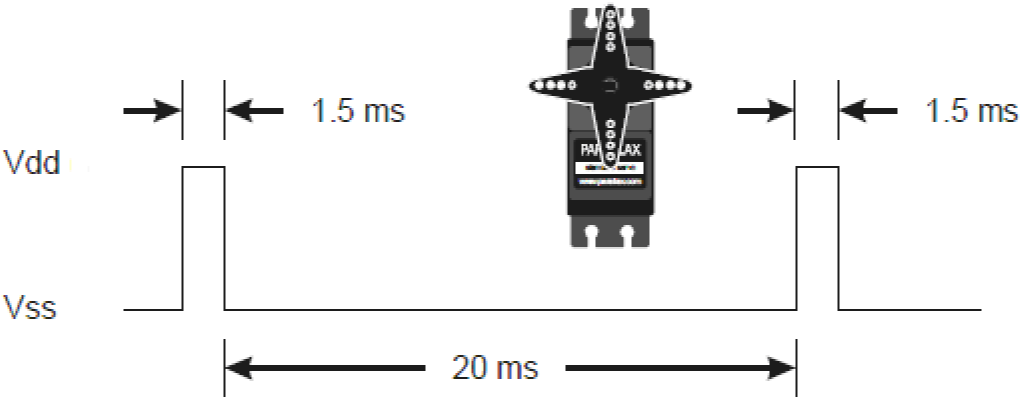
Propeller Chip Calibration Code – for P8X32A

- √ Download and unzip the Propeller code file from the 900-00008 product page.
- √ Connect the servo line to pin 0.
- √ Run the program CenterServo.spin, and gently twist the potentiometer adjustment screw until the servo does not turn or vibrate. *NOTE: Calibrating the servo may take some patience. The potentiometer is very sensitive so a very light touch will be required.*

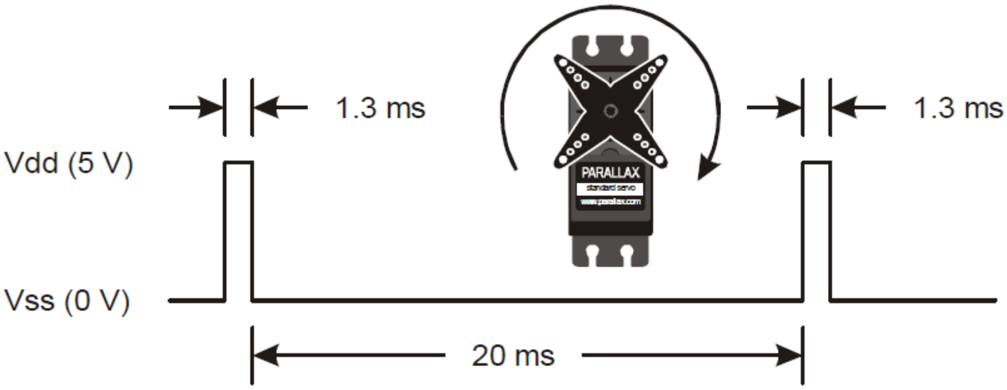
```
CenterServo.spin

CON
_clkmode = xtal1 + pll16x                     ' System clock → 80 MHz
_xinfreq = 5_000_000

PUB CenterServo | tInc, tc, tHa, t

ctra[30..26] := %00100                        ' Configure Counter A to NCO
ctra[8..0] := 0

frqa := 1
dira[0]~~

' Set up cycle and high times
tInc := clkfreq/1_000_000
tC   := tInc * 21_500
tHa  := tInc * 1500
t    := cnt                                   ' Mark counter time

repeat                                        ' Repeat PWM signal
  phsa := -tHa                                ' Set up the pulse
  t += tC                                     ' Calculate next cycle repeat
  waitcnt(t)                                  ' Wait for next cycle
```

Communication Protocol

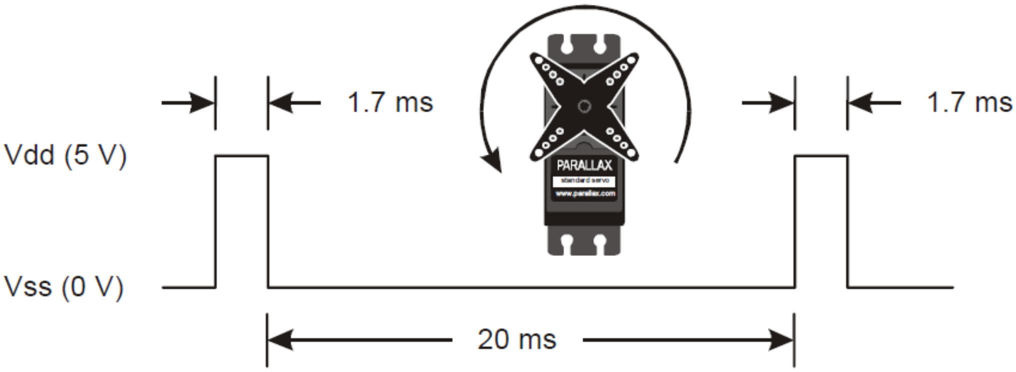
The Parallax Continuous Rotation Servo is controlled through pulse width modulation, where the rotational speed and direction are determined by the duration of the pulse. In order for smooth rotation, the servo needs a 20 ms pause between pulses. Below is a sample timing diagram for a centered servo:



As the length of the pulse decreases from 1.5 ms, the servo will gradually rotate faster in the clockwise direction, as can be seen in the figure below:



Likewise, as the length of the pulse increases from 1.5 ms, the servo will gradually rotate faster in the counter-clockwise direction, as can be seen in the figure below:



BASIC Stamp Programming Examples

PBASIC has a PULSOUT command that sets the I/O *Pin* to an output and sends a pulse of the specified *Duration*. Since the servo needs this pulse refreshed every 20 ms for continuous operation, the PULSOUT command is put in a counted FOR...NEXT loop to sustain continuous operation for the specified number of cycles.

PULSOUT *Pin*, *Duration*

Different BASIC Stamp modules use different units for the PULSOUT command's *Duration* argument. When adapting BS2 code to another BASIC Stamp model, you may need to make adjustments. The table below lists the PULSOUT ranges for each BASIC Stamp microcontroller. See the BASIC Stamp Manual or BASIC Stamp Editor Help for more information.

BASIC Stamp Module	1.3 ms	1.5 ms	1.7 ms
BS1	100	150	200
BS2, BS2e, BS2pe	500	750	1000
BS2sx, BS2px, BS2p24/40	1250	1875	2500

The example shown below for a BASIC Stamp 2 causes a servo connected to BASIC Stamp 1/0 pin 12 to first rotate full-speed clockwise for about 5 seconds, hold still for about 5 seconds, then rotate counterclockwise for 5 seconds.

```
' {$STAMP BS2}
' {$PBASIC 2.5}

counter VAR Word

FOR counter = 1 TO 100                  ' Rotate clockwise for ~5 seconds

  PULSOUT 12, 850
  PAUSE 20

NEXT

FOR counter = 1 TO 100                  ' Hold still for ~5 seconds

  PULSOUT 12, 750
  PAUSE 20

NEXT

FOR counter = 1 TO 100                  ' Rotate counterclockwise for ~5 seconds

  PULSOUT 12, 650
  PAUSE 20

NEXT
```

For more examples with the BASIC Stamp 2, including 2-wheeled robot maneuvers and ramping, see "Robotics with the Boe-Bot" Chapter 4, available for free download from the 28132 product page at www.parallax.com.

Propeller Application

The program below uses counter modules to rotate the servo first clockwise at full speed for 2 seconds, then rests for 2 seconds, and rotates counterclockwise at full speed for another 2 seconds. This code can also be downloaded from the 900-00008 product page.

```
ServoRotation.spin

CON
_clkmode = xtal1 + pll16x                ' System clock → 80 MHz
_xinfreq = 5_000_000

PUB CenterServo | tInc, tc, tCtr, tCw, tCcw, t

ctra[30..26] := %00100                   ' Configure Counter A to NCO
ctra[8..0] := 0

frqa := 1
dira[0]~~


tInc := clkfreq/1_000_000                ' 1 µs increment
tC   := tInc * 21_500                    ' Low pulse
tCtr := tInc * 1500                      ' Center pulse = 1.5 ms
tCw  := tInc * 1300                      ' Clockwise fast = 1.3 ms
tCcw := tInc * 1700                      ' Counter-Clockwise fast = 1.7 ms
t    := cnt                              ' Mark counter time

repeat 100                               ' Repeat PWM signal 100x
  phsa := -tCw                           ' Set up clockwise fast pulse
  t += tC                                ' Calculate next cycle repeat
  waitcnt(t)                             ' Wait for next cycle (20 ms)

repeat 100                               ' Repeat PWM signal 100x
  phsa := -tCtr                          ' Set up the center pulse
  t += (tC + 200)                        ' Calculate next cycle repeat
  waitcnt(t)                             ' Wait for next cycle (20 ms)

repeat 100                               ' Repeat PWM signal 100x
  phsa := -tCcw                          ' Set up counter-clockwise fast pulse
  t += (tC - 200)                        ' Calculate next cycle repeat
  waitcnt(t)                             ' Wait for next cycle (20 ms)
```