# HexCrawler

## Kit Assembly, Tuning and Example Program

VERSION 3.1

PARALLAX INC

Robotic CrustCrawler Design & Development

## WARRANTY

Parallax, Inc. warrants its products against defects in materials and workmanship for a period of 90 days. If you discover a defect, Parallax will, at its option, repair, replace, or refund the purchase price. Simply call for a Return Merchandise Authorization (RMA) number, write the number on the outside of the box and send it back to Parallax. Please include your name, telephone number, shipping address, and a description of the problem. We will return your product, or its replacement, using the same shipping method used to ship the product to Parallax.

## 14-DAY MONEY BACK GUARANTEE

If, within 14 days of having received your product, you find that it does not suit your needs, you may return it for a full refund. Parallax will refund the purchase price of the product, excluding shipping / handling costs. This does not apply if the product has been altered or damaged.

## COPYRIGHTS AND TRADEMARKS



## DISCLAIMER OF LIABILITY

Parallax, Inc. is not responsible for special, incidental, or consequential damages resulting from any breach of warranty, or under any legal theory, including lost profits, downtime, goodwill, damage to or replacement of equipment or property, and any costs or recovering, reprogramming, or reproducing any data stored in or used with Parallax products. Parallax is also not responsible for any personal damage, including that to life and health, resulting from use of any of our products. You take full responsibility for your BASIC Stamp application, no matter how life-threatening it may be.

## INTERNET ACCESS

We maintain internet systems for your use. These may be used to obtain software, communicate with members of Parallax, and communicate with other customers. Access information is shown below:

E-mail: info@parallax.com
Web: http://www.parallax.com

## INTERNET BASIC STAMP DISCUSSION LIST

We maintain two e-mail discussion lists for people interested in BASIC Stamp microcontrollers (subscribe at http://www.parallax.com under the technical support button). The BASIC Stamp list server includes engineers, hobbyists, and enthusiasts. The list works like this: lots of people subscribe to the list, and then all questions and answers to the list are distributed to all subscribers. It’s a fun, fast, and free way to discuss BASIC Stamp issues and get answers to technical questions. This list generates about 40 messages per day. The Stamps in Class list is for students and educators who wish to share educational ideas. To subscribe to this list go to http://www.parallax.com/sic and look for the E-groups list. This list generates about 5 messages per day.

# Table of Contents

# Preface

The HexCrawler is an original design from Alex Dirks of CrustCrawler (www.crustcrawler.com). Parallax is the exclusive distributor and final kit assembler of the product. Parallax Inc. has taken many efforts to ensure that the kit includes only the highest quality metal parts, electronics and instructions to ensure that you get the HexCrawler operating correctly. If you are not having success with the kits or just have some feedback, send it to us.

The HexCrawler hexapod is an advanced robotic kit consisting of a walking platform. Unlike other Parallax robotic kits, this kit does not include a series of step-by-step educational projects for the HexCrawler. Essentially, the kit is a platform for your additional robotic projects.

The applications Parallax provides on our HexCrawler web pages are project-oriented. For example, in October 2003 we completed an application with a video camera and remote control for telerobotic applications. This application is available for download.

Experience programming BASIC Stamp microcontrollers is helpful, but if you need more help in this regard you can find plenty of robotic programming resources on the Parallax web site. All of the sensors we sell include BASIC Stamp program examples which you could readily adapt to the HexCrawler.

Putting the hardware together also requires some skill with hand tools. If you are not semi-skilled with common hand tools we recommend you return the kit prior to assembly unless you have some patience.

But, the Parallax team assures you that if you can successfully complete the HexCrawler you're in for an exciting series of robotic projects that you will find highly rewarding. Our office staff has taken to the HexCrawler by customizing it with unique paint jobs, cameras and ultrasonic sensors.

# Chapter #1: Preparing to Assemble the HexCrawler

1

## REQUIRED TOOLS

The following tools will be required to build your HexCrawler:

- Phillips screwdriver
- Drill
- 1/8” Drill Bit
- Small adjustable crescent wrench or socket set
- Wire Cutters
- A small amount of white grease or equivalent

## HEXCRAWLER FULL KIT INVENTORY (#30063)

The HexCrawler Complete Kit (#30063) contains the following components:

### Electronics:

- (1) BASIC Stamp® 2 Module
- (1) Board of Education® (BOE) carrier board
- (12) HiTec HS-322 HD Servos
- (1) serial cable
- (1) Parallax CD-ROM
- (1) Parallax Servo Controller (PSC)
- (1) seven-segment LED (green)
- (2) 220 Ω resistors (red, red, brown)
- (7) 1 kΩ resistors (red, black, brown)
- (2) 10 kΩ resistors (brown, black, orange)
- (2) pushbuttons
- (1) package 3” jumper wires

### Aluminum Parts:

- (1) upper main body deck
- (1) lower main body deck
- (1) square rear support brace
- (1) flared front support brace
- (5) Support brackets
- (12) lower, horizontal leg braces
- (12) vertical leg braces
- (6) leg actuators
- (6) leg actuator supports
- (6) servo mounts
- (12) lower leg braces - (6 ) short, (6) Long

## Nuts, Bolts, Washers and Screws

- (12) #2 nuts
- (12) #2 lock washers
- (12) #2 washers
- (20) #4 washers
- (20) #4 -1/2” screws
- (40) #4 -5/16” screws
- (14) #4 nuts
- (18) #4 lock nuts
- (6) #4 ¼” nylon spacers
- (8) #4 3/16” nylon spacers
- (24) #6 -3/8” screws
- (24) #6 lock nuts
- (60) #8 flat washers
- (6) #8 -1” screws
- (30) #8 lock nuts
- (24) #8- 1.25” screws
- (12) #8 -½” nylon spacers
- (18) #8 -¼” nylon spacers
- (36) #8 flat nylon spacers
- (12) #8- 7/16” nylon spacers
- (6) ¼” SAE flat washers

## Miscellaneous

- HexCrawler Manual
- (6)- ¾” long - 2/56 threaded rods
- (6) rubber feet
- (12) dog bones
- (12) ball links
- (12) cable ties
- (4) 12” servo wire exensions

## Source Code from the Parallax Web Site (www.parallax.com)

### HexCrawler Source Code

The Parallax web site HexCrawler page contains sample BS2-IC source code, this installation guide, and additional pictures of the HexCrawler to aid you during the construction process. Also, we frequently post additional applications for our products on the web site. For example, in October 2003 we posted a HexCrawler application with a video camera and remote control. Crustcrawler (www.crustcrawler.com) also provides an extensive listing of BS2-IC source code for the HexCrawler.

The Parallax CD-ROM includes the *BASIC Stamp Manual* and many valuable resources to assist you with your BASIC Stamp programming efforts.

### HexCrawler Projects / Accessories

Projects that include updated code, electronics and hardware accessories are always being added to the Parallax and Crustcrawler web sites. Check with our sites often for the latest updates.

## ADDITIONAL PARTS YOU NEED TO SUPPLY

Like other hobby kits, completing the HexCrawler kits requires additional parts that you will need to supply. This hardware is not included in the Parallax kits because it would only drive the cost higher as most of these items are readily available at hobby stores or www.towerhobbies.com (purchasing them from Parallax, if available, would probably have a higher price).

- 7.2 V NiMH or NiCd six-cell rechargeable battery for servo power. This is a standard 1800 mAH to 3000 mAH battery pack, commonly used in R/C cars. A good source for this product is Tower Hobbies (www.towerhobbies.com). Expect to pay between $30 and $60. Or, you can go to Radio Shack who also sells good quality 7.2V NiMH batteries and chargers.

**Note:** Servos perform best when using "regulated" power from either a fixed 5v regulated power supply or a 5v, voltage regulator circuit connected between your battery (minimum of 6 volts) and the Parallax Servo Controller's servo power inputs. Using unregulated power from a battery pack may result in some twitching of the servos when the HexCrawler is idle or not moving. This is perfectly normal and does not harm the servos. For users who want "twitch free" performance from their HexCrawlers, Crustcrawler (www.crustcrawler.com) sells a voltage regulator assessory pack and lists a schematic for the Hexcrawler in the electronics sections of their web site.

- AC/DC Digital Peak Charger for the 7.2 V NiCd/NiMH battery. One acceptable product is the Piranha Digital Charger from Tower Hobbies (their stock code #LXCLD5). Probably around $50 or less. Radio Shack also sells chargers for their R/C cars.
- Zip-ties of the smaller sizes are very useful as cable ties for securing wires in a tidy fashion. Approximate cost is probably a few dollars. Available from any hardware store. (12) cable ties have been included in your kit.
- Robot sensors The HexCrawler is a platform and the opportunities for sensor integration are endless. Selecting the appropriate sensor is left up to you, our customer. Parallax and CrustCrawler (www.crustcrawler.com) have many acceptable add-ons for the CrustCrawler. There is no cost limit in this regard.

# Chapter #2: Pre-Assembly Tips



## PAY ATTENTION TO DETAILS

- Work in a well lit, clean environment with lots of workspace
- Obtain a small stack of books that are approximately the length and width of the HexCrawler's lower deck and is approximately 3.5" to 4" tall when stacked. The added height will be required when attaching the lower and upper decks together.
- Organize your nuts, bolts and screws so that you have each specific size of lock-nuts, screws, and washers in the same group, and place them so they are easily within reach.
- Take your time! The HexCrawler kit is a precision-made product that contains a lot of parts and requires all parts to be assembled in the *exact* order as described in this installation manual.
- The average time to construct a kit is between 5 and 8 hours.
- Additional pictures of the construction process may be downloaded from the Parallax web site's HexCrawler pages. During the construction process, please refer to these pictures and the ones in this guide frequently as a reference (http://www.parallax.com/detail.asp?product_id=30063).
- Refer to and study the pictures and close-up diagrams carefully **before** starting the construction of any part of your kit.
- Always note the **orientation** and **direction** of screws and aluminum parts and which **side** of your HexCrawler you are constructing! It absolutely makes a difference!
- The code and procedures in this guide have been extensively tested and verified for accuracy. If you find that something is not working properly, be sure to re-check ALL of your wiring and electronics connections against the schematics and illustrations in this manual first.

## PREPARING THE SERVOS

- Remove the aluminum body parts from their protective bags and lay them loosely in their respective groups on your work surface.
- Gather (6) of the servos and remove the servo horn and screw, setting them aside in a safe place.
- Remove the riser tabs from both sides of (6) of the servos as shown in Figure 1. The removal of this plastic riser will allow the servo to sit flush against the servo holder. These servos will be installed in the next few steps of the construction process.

# Chapter #3: HexCrawler Assembly



## MOUNTING THE SERVOS

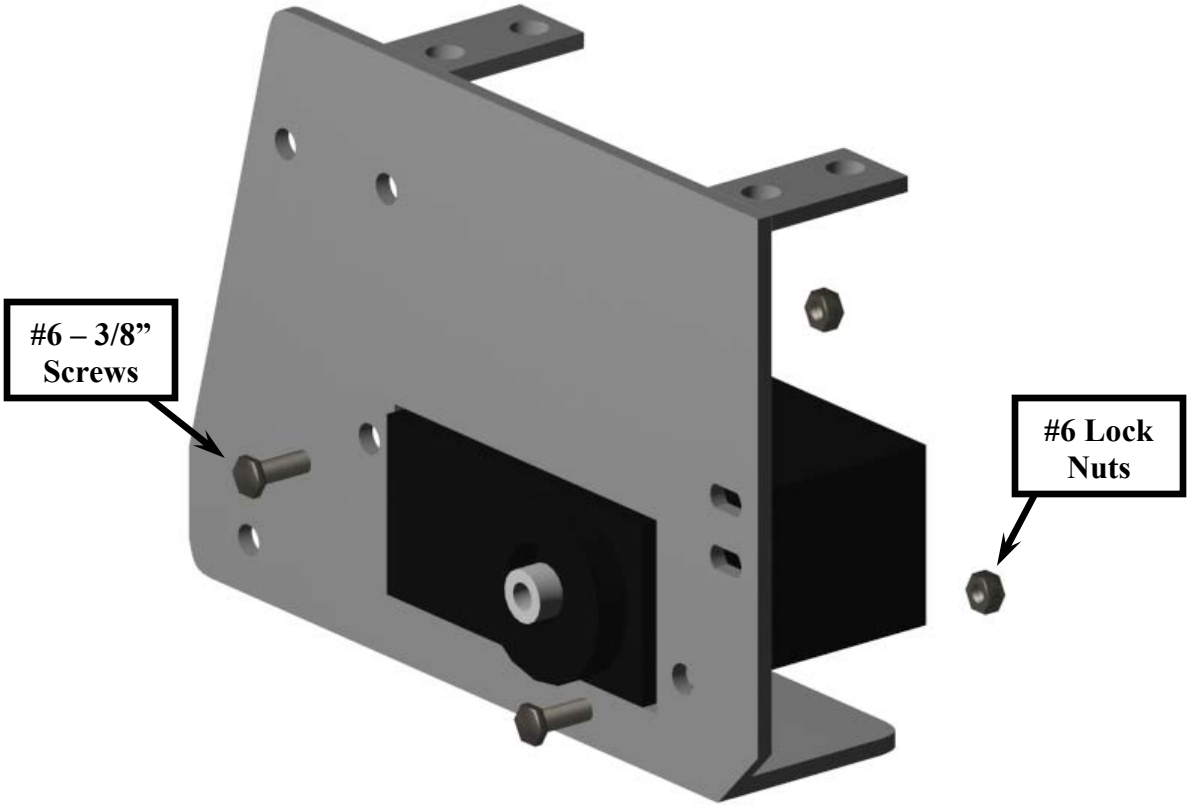



**Figure 2: Mounting Servos**

1. Install the servo into the rectangular portion of the servo mount. The servo gear head should always be orientated towards the straight end of the servo mount as shown in Figures 2 and 3.

2. Using (2) #6 - 3/8" screws and lock nuts, install the servo to the aluminum servo assembly. Complete this step for all (6) legs of your HexCrawler.


**Figure 3: Servo Orientation**

3. Using (2) #6 - 3/8" screws and lock nuts, install the top servo to the aluminum servo assembly as shown in figure 4. Ensure that the servo is mounted <u>on top</u> of the servo tabs as shown in Figure 4, 5, and 6. Complete this step for all (6) legs of your HexCrawler.

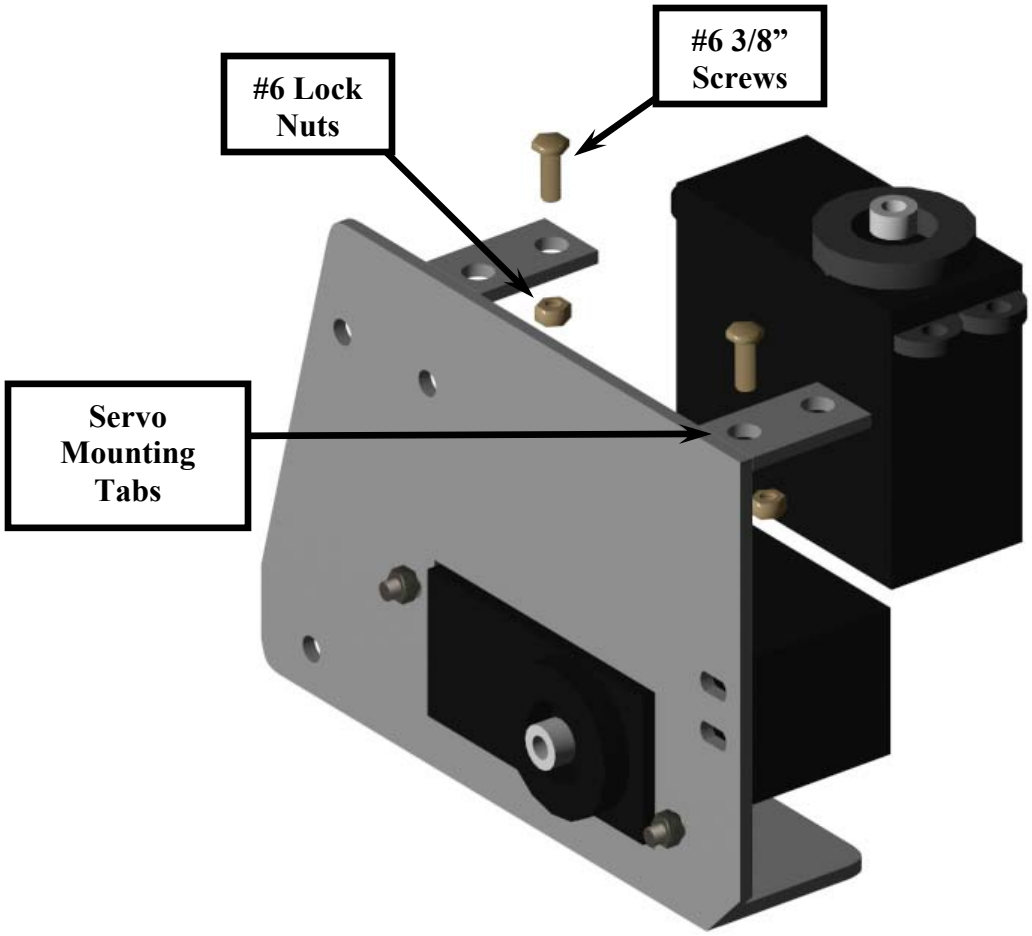



**Figure 4**: Top Servo Installation

**Figure 5:** Completed Servo Installation – Front view

**Figure 6:** Completed Servo Installation – Rear view

4. Install the Du-Bro 2-56 threaded ball link to all (6) of the main leg actuators as shown in Figure 7 below. Ensure that the ball link is centered in the slot.

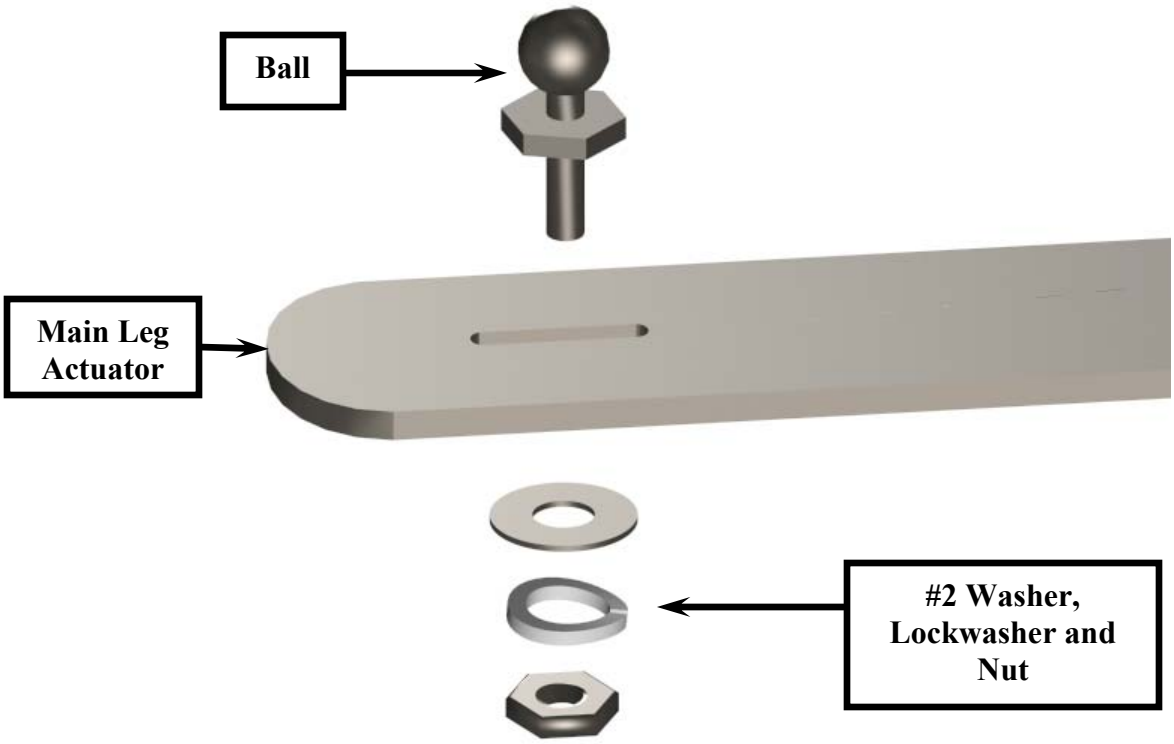



**Figure 7: Leg Actuator Assembly**

5. Steps 6 - 14 should be performed for each leg in your kit. Once one leg has been assembled, use the finished leg as a quick reference for the assembly of the rest of the legs.

6. Using (1) #8, 1.25" screw, install the leg actuator to the leg actuator support through the #8 hole near the top of the servo mount as shown in Figure 8. Use the same hole for the right and left leg assemblies.

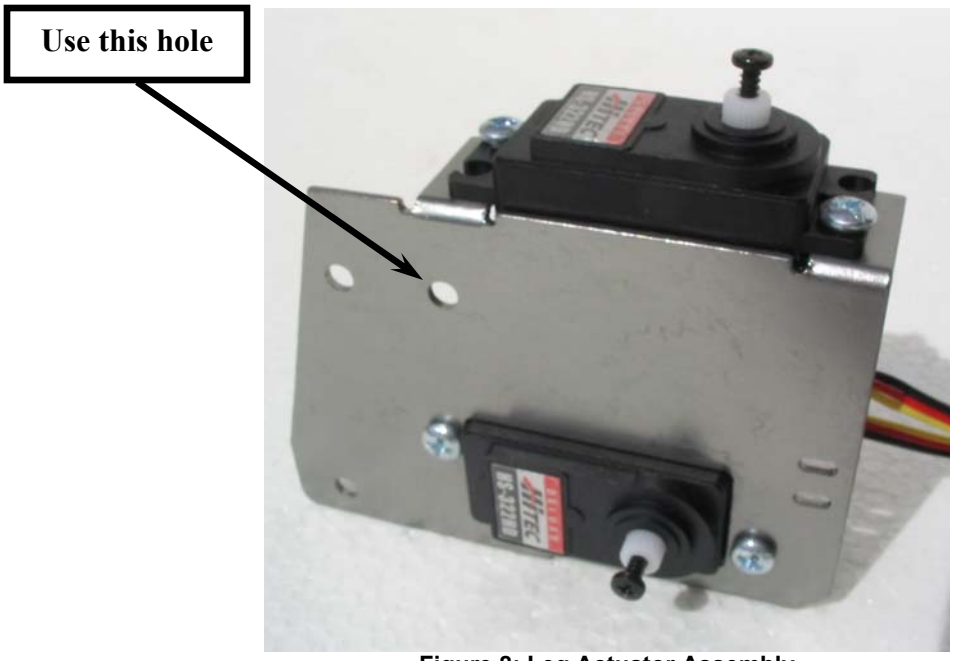



**Figure 8: Leg Actuator Assembly**

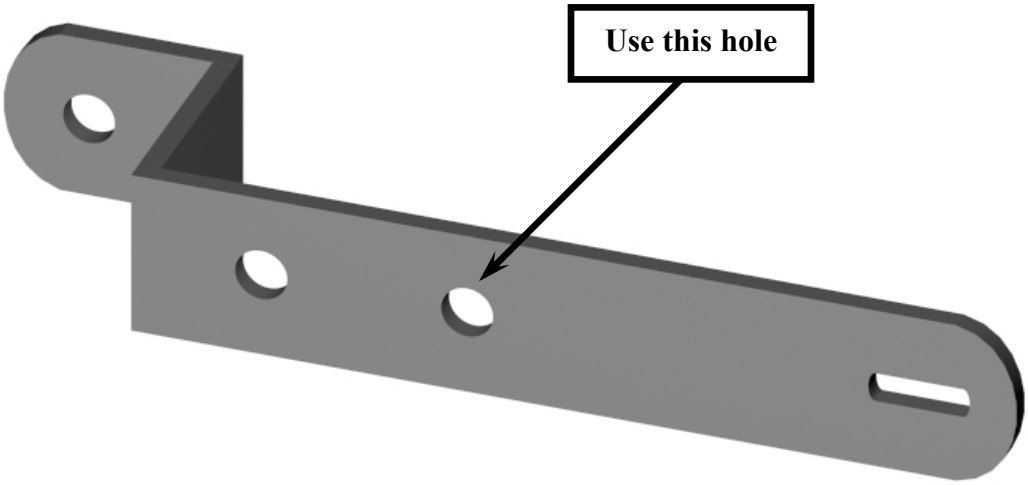



**Figure 9: Leg Actuator**

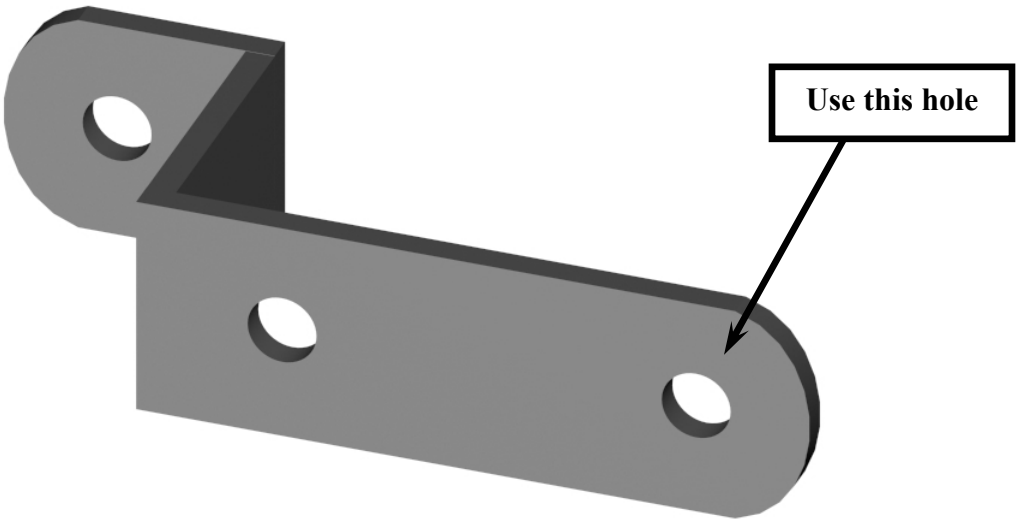



**Figure 10: Leg Actuator Support**

7. Use the close-up diagram in Figure 11 for the complete assembly sequence.

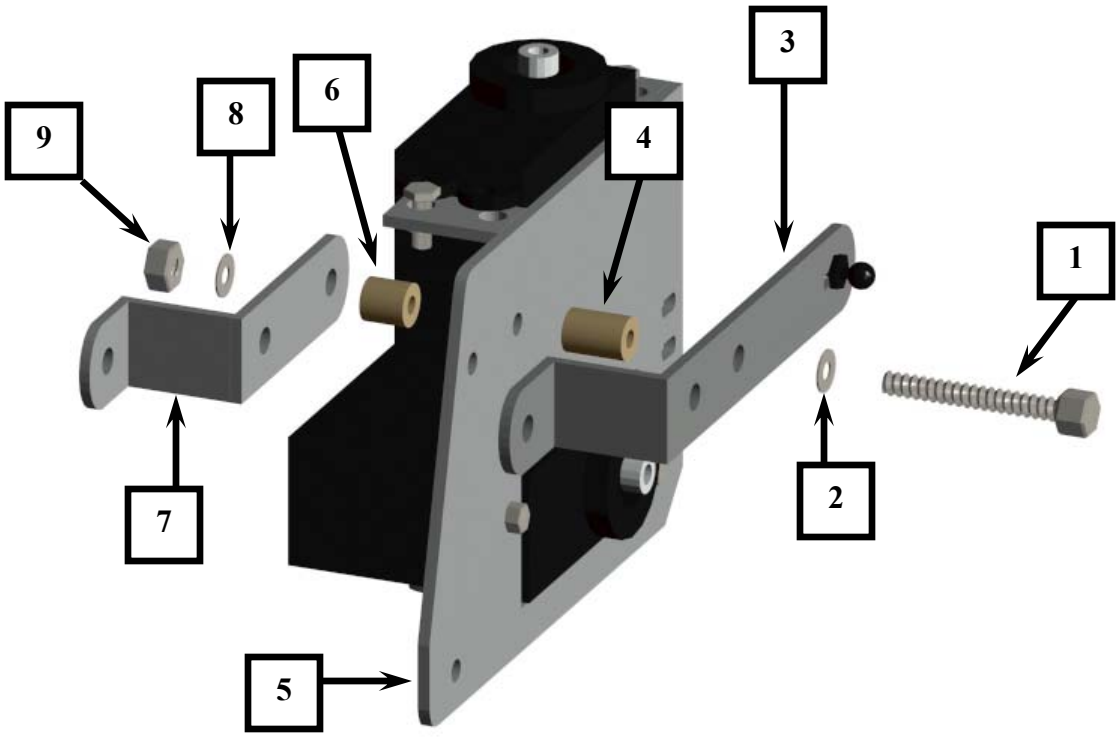



**Figure 11: Leg Actuator Assembly**

| | |
|---|---|
| 1 - #8 1.25” Screw | 6- #8 ¼” Nylon Spacer |
| 2 – #8 Stainless Steel Washer | 7 – Leg Actuator Support |
| 3 – Leg Actuator | 8 – #8 Stainless Steel Washer |
| 4 – #8 7/16” Nylon Spacer | 9 – #8 Lock Nut |
| 5 – Servo Mount | |

8. At this point in the process, the leg assembly should look like Figures 12 and 13. Note the location and orientation of the holes and ball socket slot. Tighten the screw and lock nut just enough so that the 2 pieces can be easily moved by hand but rigid enough to stand on their own. These screws will be loosened later in the construction process to ensure smooth, friction free leg movement. Do not over-tighten the screws as it makes the rest of the leg assembly difficult to complete.

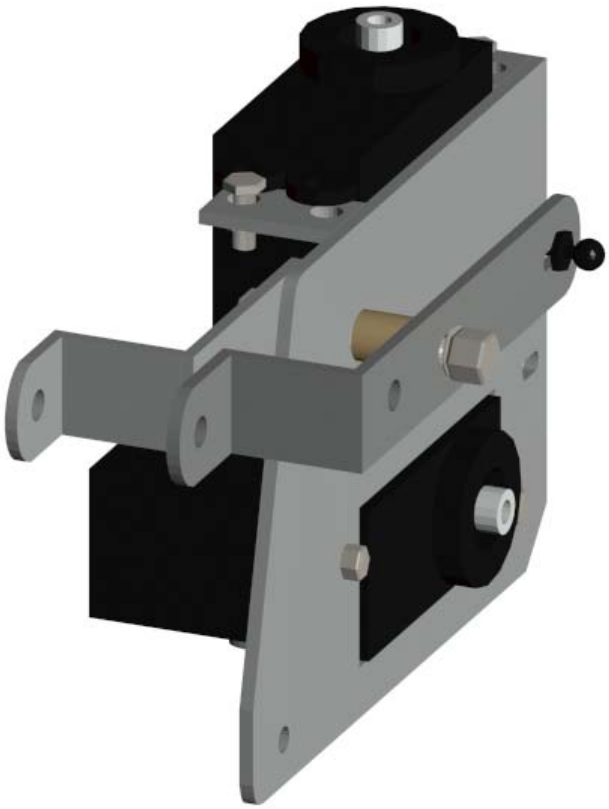


**Figure 12: Leg Actuator Assembly**

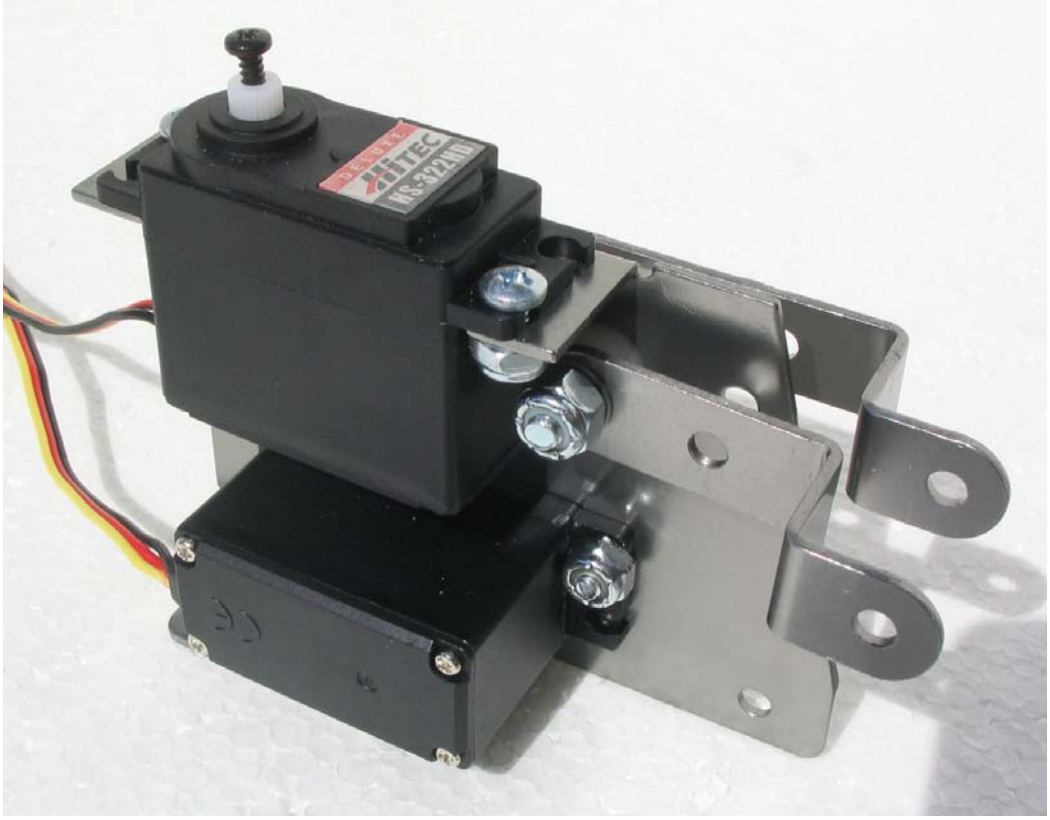



**Figure 13:** The Completed Leg Actuator Assembly

9. Attach (2) lower horizontal leg braces with the same hardware and in the same order as the previous steps to the lower hole on the servo mount as shown in Figure 14. Use the close-up diagram on the next page as a reference. Ensure that the shorter end of the lower horizontal leg brace is installed to the servo mount as shown in Figure 14 and Figure 15.

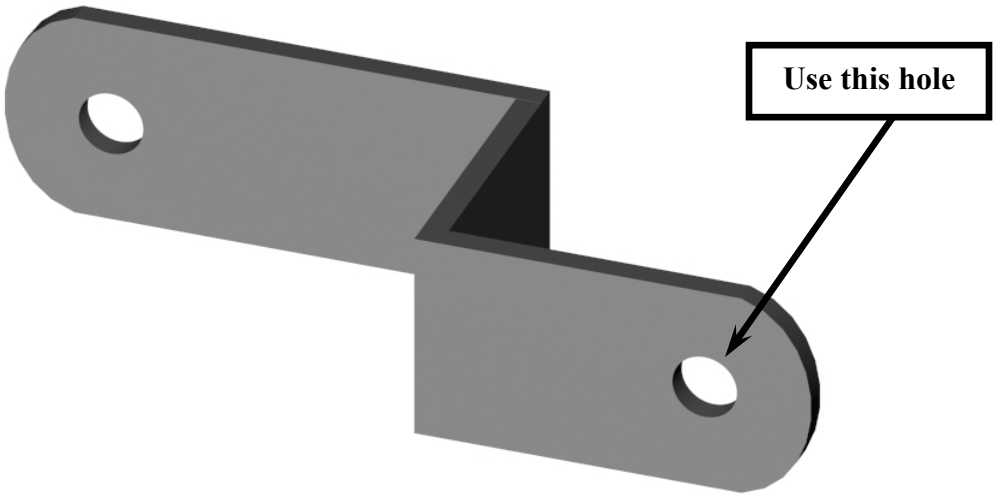



**Figure 14: Lower Horizontal Leg Brace**

9. Use the close-up diagram in Figure 15 for the complete assembly sequence.

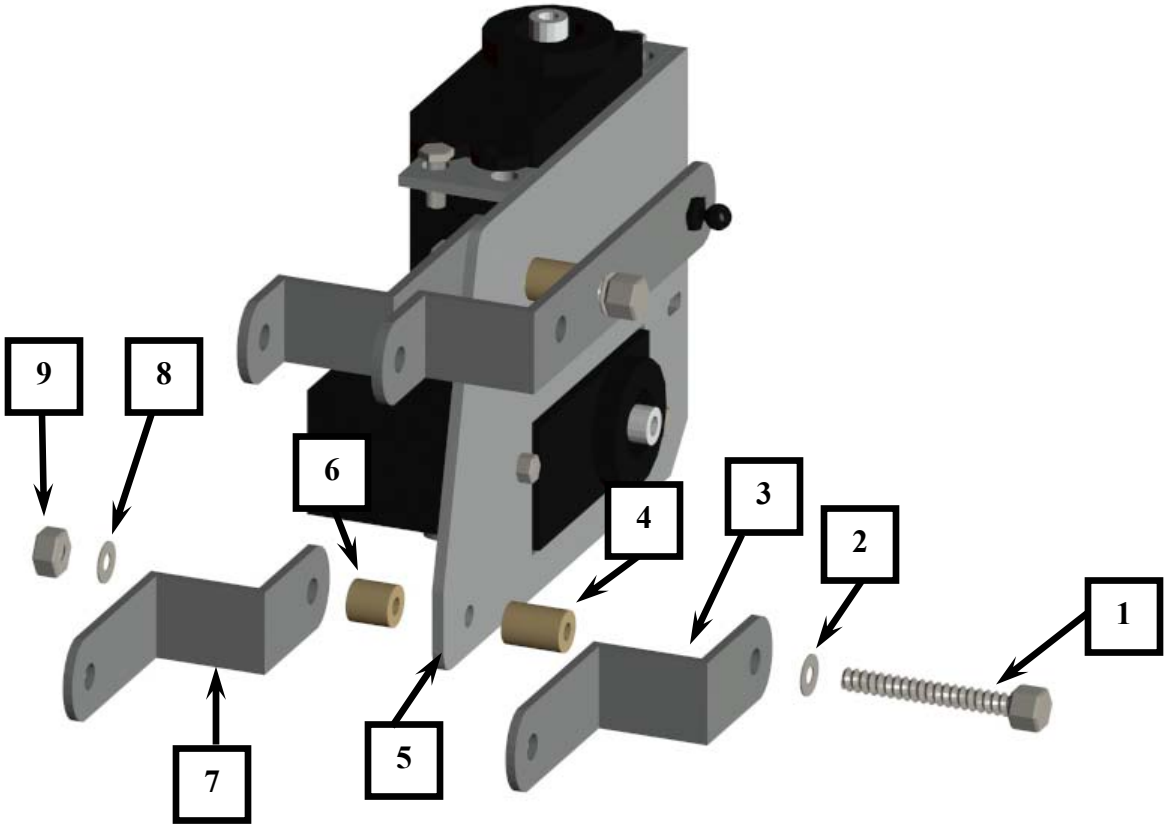



**Figure 15: Lower Horizontal Leg Assembly**

| | |
|---|---|
| 1 - #8 Screw 1-1/4” | 6- #8 ¼” Nylon Spacer |
| 2 – #8 Stainless Steel Washer | 7 – Lower Horizontal Leg Brace |
| 3 – Lower Horizontal Leg Brace | 8 – #8 Stainless Steel Washer |
| 4 – #8 7/16” Nylon Spacer | 9 – #8 Lock Nut |
| 5 –Servo Mount | |

When completed, the leg assembly should look like Figure 16 and Figure 17.

**Figure 16**

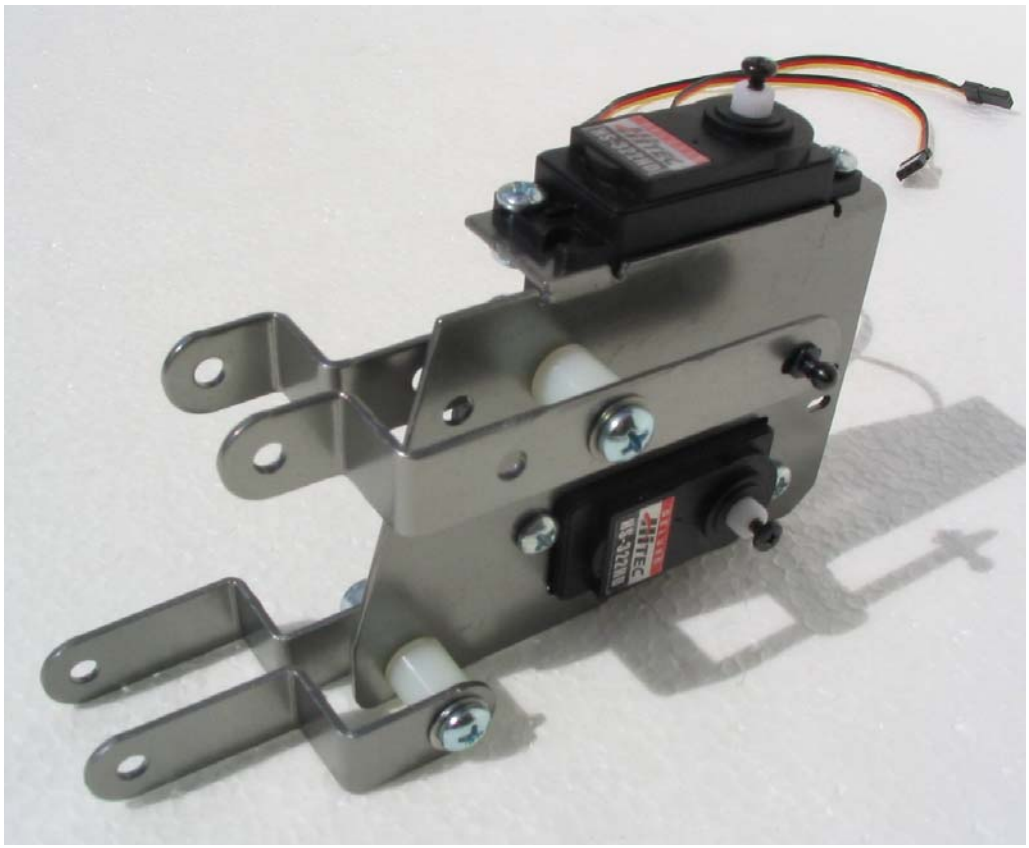


**Figure 17**

10. Next, install the 2 vertical leg braces (Figure 18) to the leg assembly. Use Figure 19 to assemble the vertical leg braces. Note that the top vertical brace assembly is identical to the lower vertical brace assembly.

**Figure 18: Vertical Leg Brace**

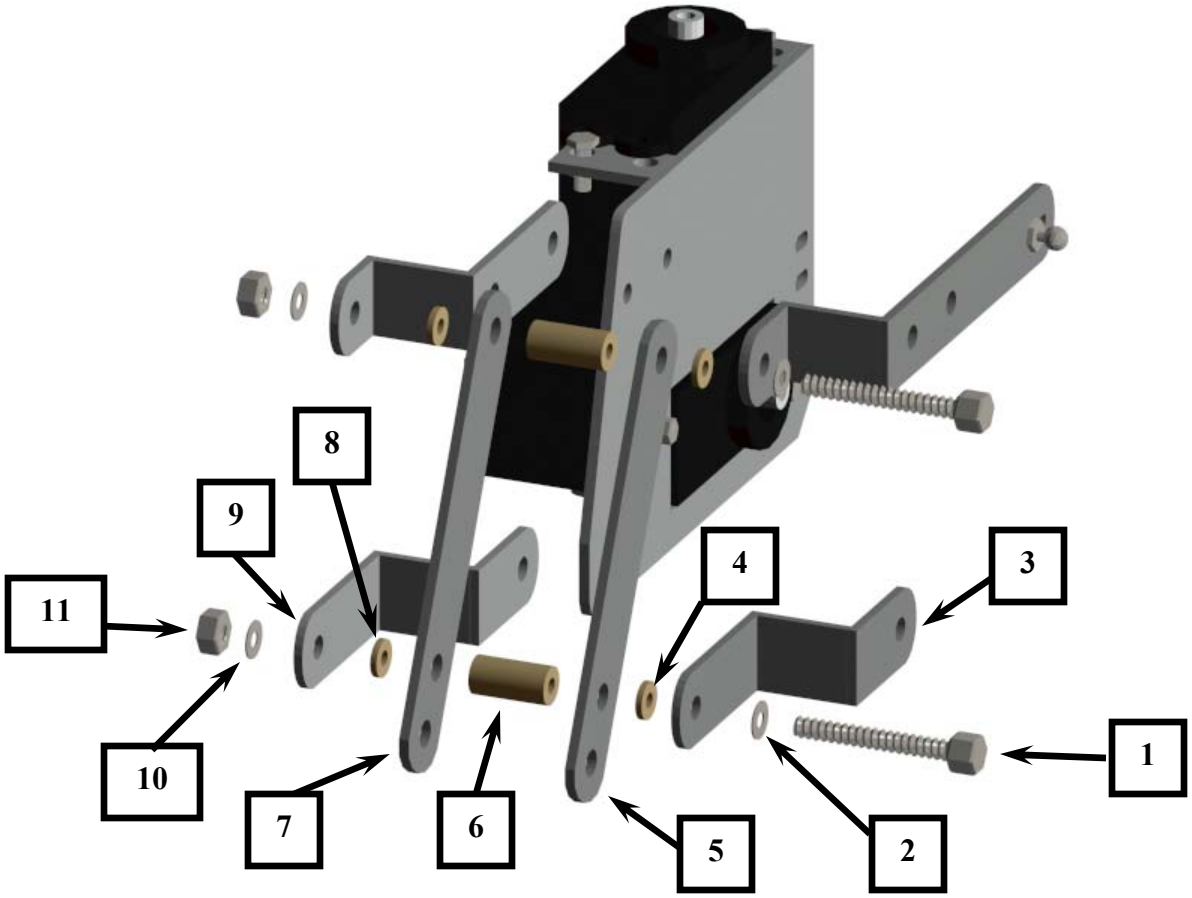



**Figure 19: Vertical Leg Assembly**

| | |
|---|---|
| 1 - #8 1 1/4” Screw | 7 - Vertical Leg Brace |
| 2 - #8 Stainless Flat Washer | 8 - #8 Flat Nylon Spacer |
| 3 - Lower Horizontal Leg Brace | 9 - Lower Horizontal Leg Brace |
| 4 - #8 Flat Nylon Spacer | 10 - #8 Stainless Flat Washer |
| 5 - Vertical Leg Brace | 11 - #8 Lock Nut |
| 6 - #8 ½” Nylon Spacer | |

| | |
|---|---|
|  | **Tip** Assemble the top of the vertical leg brace assembly first and then assemble the lower vertical leg assembly. As shown in the close-up view in figure 19, the assembly sequence and parts are identical. Do not completely tighten the upper or the lower #8 screws. Leave the screws just loose enough so that the leg assembly can be moved up and down easily by hand. |

When completed, the leg assembly should look like Figure 20.


**Figure 20: Completed Leg Assembly**

## ASSEMBLING THE LOWER LEG AND ATTACHING IT TO THE UPPER LEG ASSEMBLY

11. Use Figure 21 to assemble the lower leg assembly. Please note the orientation of the longer leg brace and the shorter brace. The shorter leg brace is always installed on top of the longer leg brace. To make construction easier, follow the numbered steps as outlined in Figure 21.

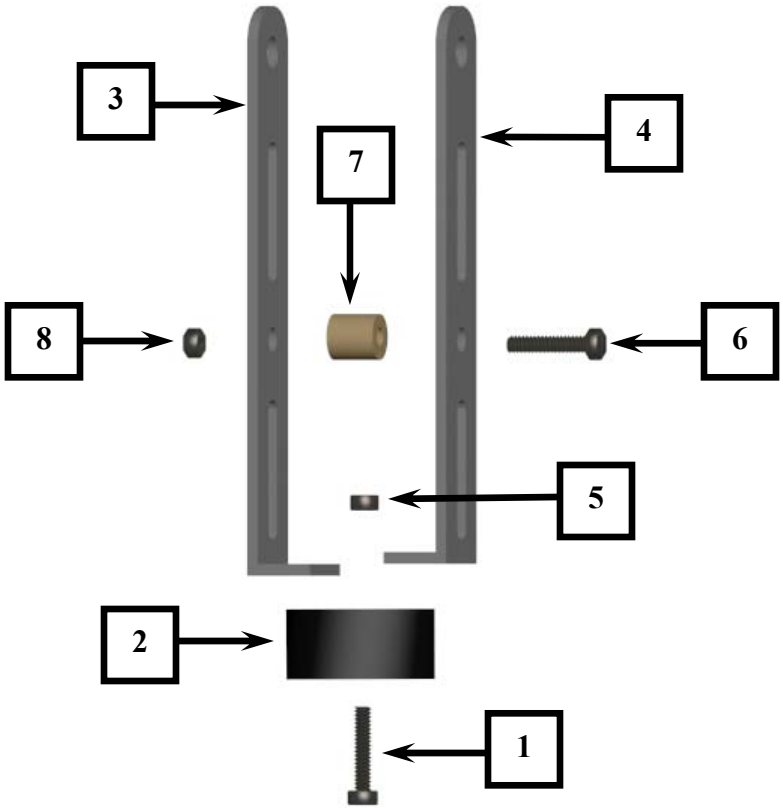



**Figure 21: Lower Leg Assembly**

| | |
|---|---|
| 1- #4 1/2” Screw | 5- #4 Nut |
| 2- Rubber Foot | 6- #4 ½” Screw |
| 3- Long Leg Brace | 7- #4 ¼” Nylon Spacer |
| 4- Short Leg Brace | 8 - #4 Lock Nut |

When completed, the leg assembly should look like Figure 22.

**Figure 22:** Completed Lower Leg Assembly

12. Attach the lower leg assembly to the upper leg assembly as shown in figure 23.

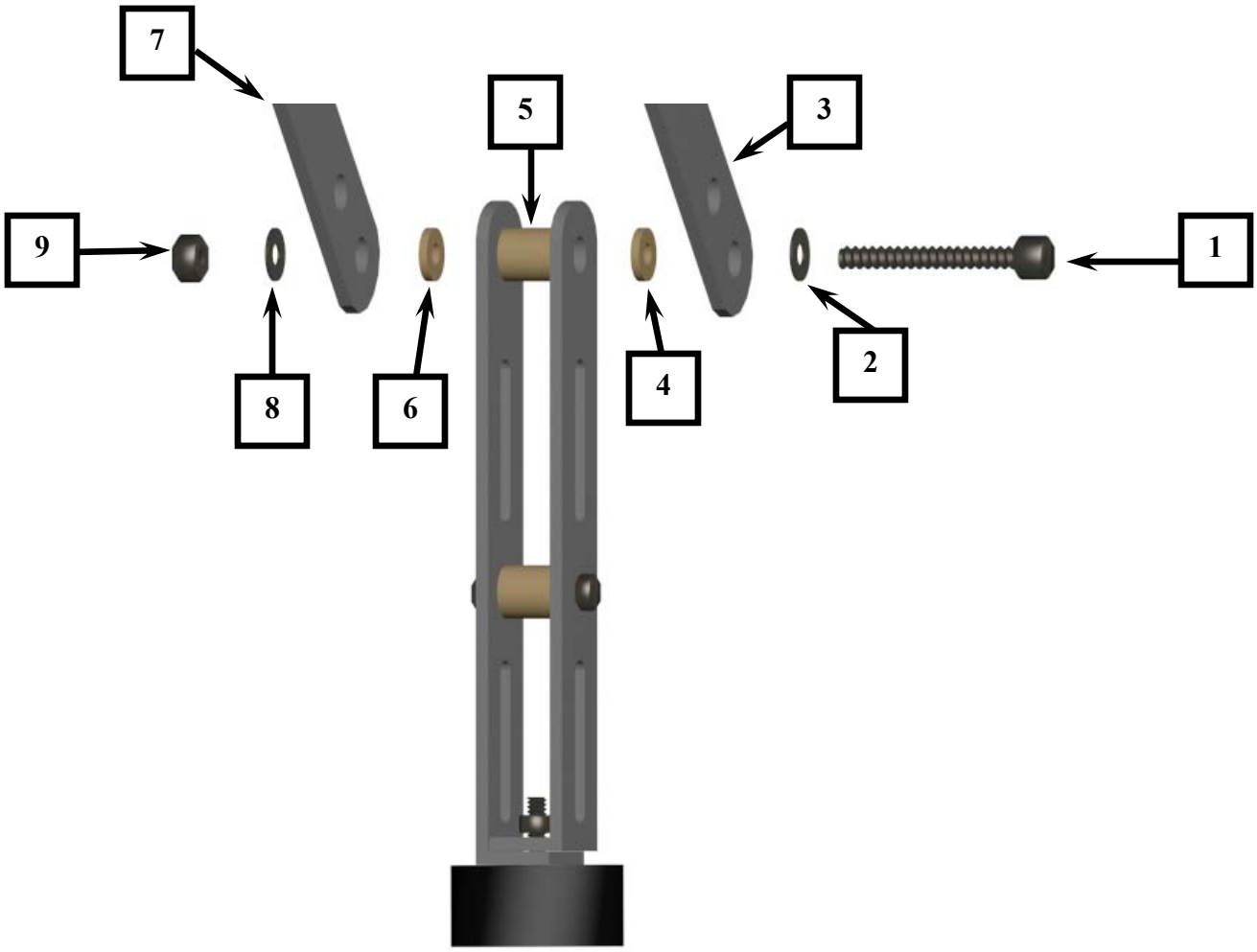



**Figure 23**

| | | | | |
|---|---|---|---|---|
| 1. #8 – 1" Screw | 3. Vertical Leg Brace | 5. #8 – ¼" Nylon Spacer | 7. Vertical Leg Brace | 9. #8 Lock Nut |
| 2. #8 Stainless Washer | 4. #8 Flat Nylon Spacer | 6. #8 Flat Nylon Spacer | 8. #8 Stainless Washer | |

When completed, the leg assembly should look like Figure 24.


**Figure 24:** Completed Lower Leg Assembly

13. Using the supplied black tie wraps, secure the 2 servo wires to the servo mount as shown in Figure 25 and Figure 26. Be sure not to stretch or extend the servo wire when securing them to the servo mount.

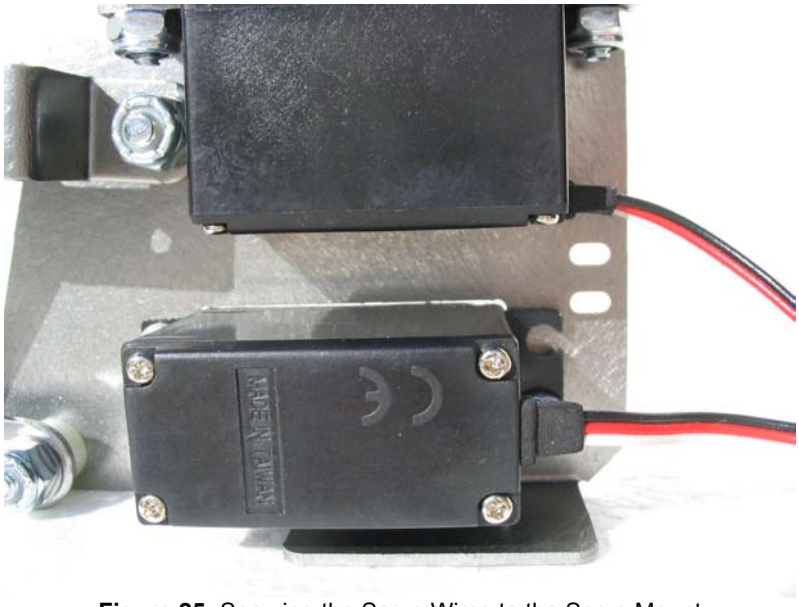


**Figure 25:** Securing the Servo Wires to the Servo Mount

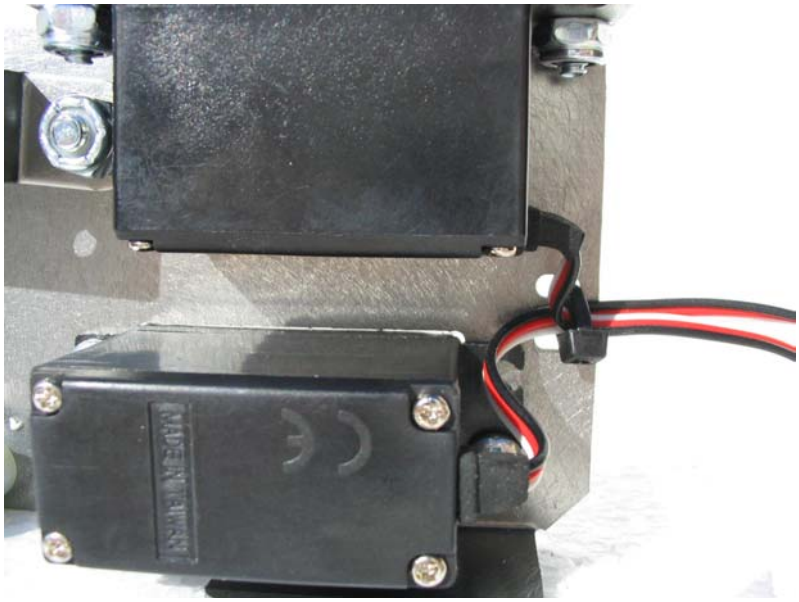


**Figure 26:** The Completed Assembly

## PREPARING THE HEXCRAWLER'S UPPER DECK

The upper deck can be identified by the 3 holes at the ends of each of the leg decks (1 large hole and 2 smaller holes). Locate the upper deck and have it ready.

1. Drill out the holes in the round servo arm as illustrated in figure 27 with a 1/8" drill bit . To make this process easier, temporarily place the servo arm on an available servo spindle to secure the arm while it is being drilled.

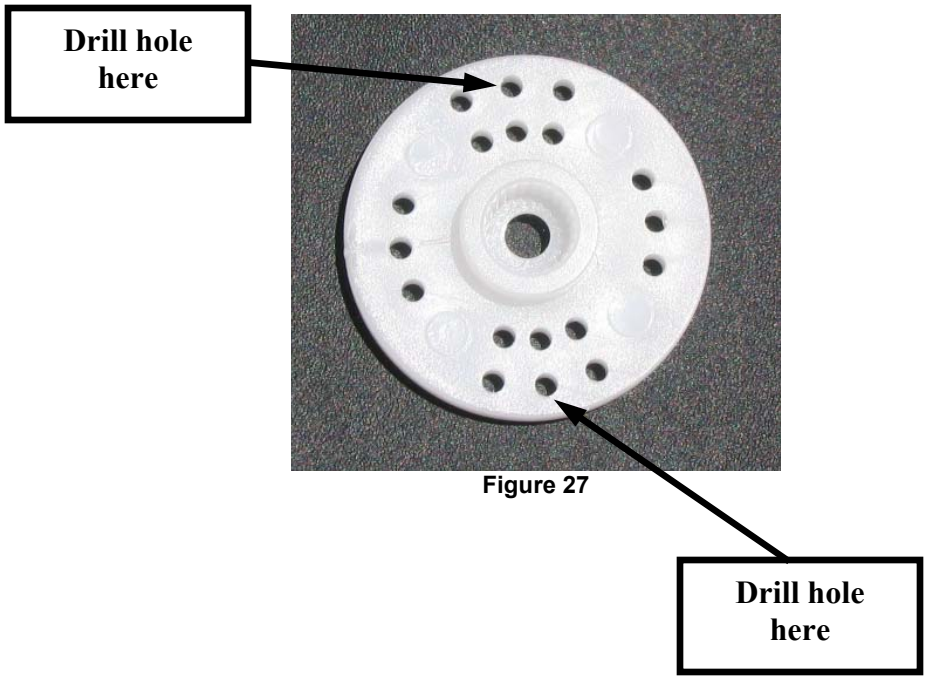



**Figure 27**

2. Using (2) 5/16” #4 screws, washers and lock nuts, install the servo arm to the upper deck ensuring that the flat side is mounted to the upper deck as shown in Figure 28.

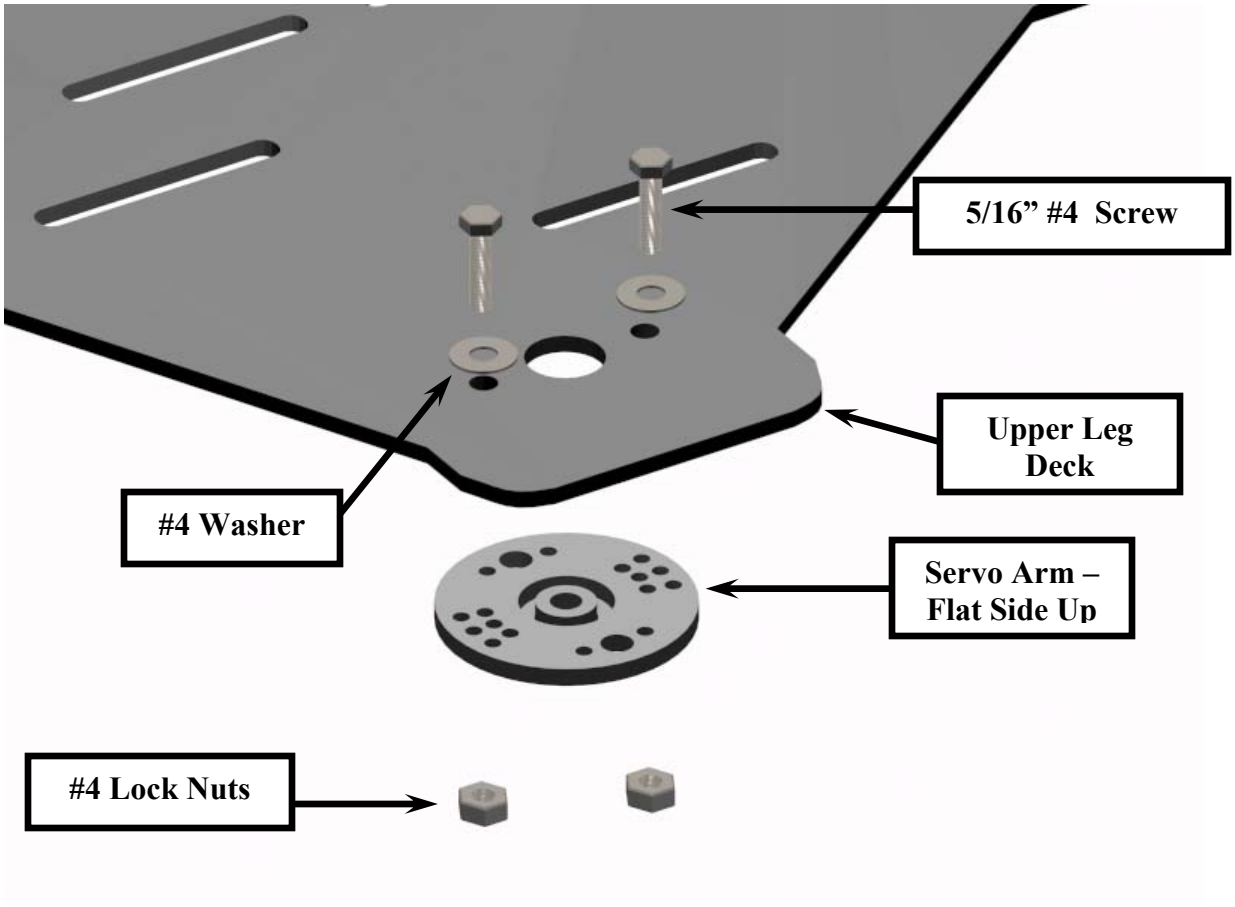



**Figure 28: Upper Deck – Servo Arm Installation**

3. Repeat step #1 and #2 for all of the leg decks. When completed, the upper deck should look like Figure 29.

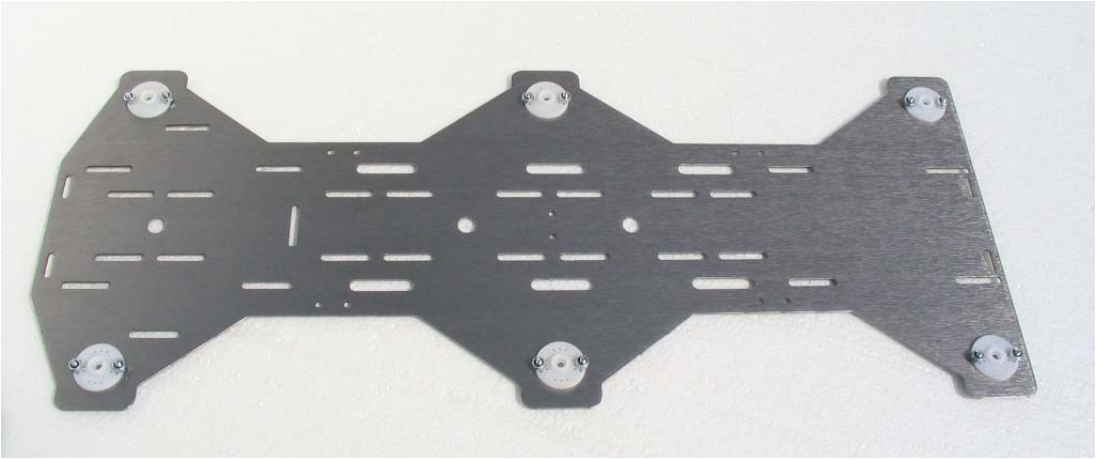


**Figure 29: Completed Upper Deck**

## INSTALLING THE SIDE AND CENTER SUPPORT BRACKETS

4. Using (2) 5/16” #4 screws, install the (4) support brackets to the side of the upper deck as shown in Figure 30. Mount (1) support bracket to the center of the upper deck as shown in Figure 31b.

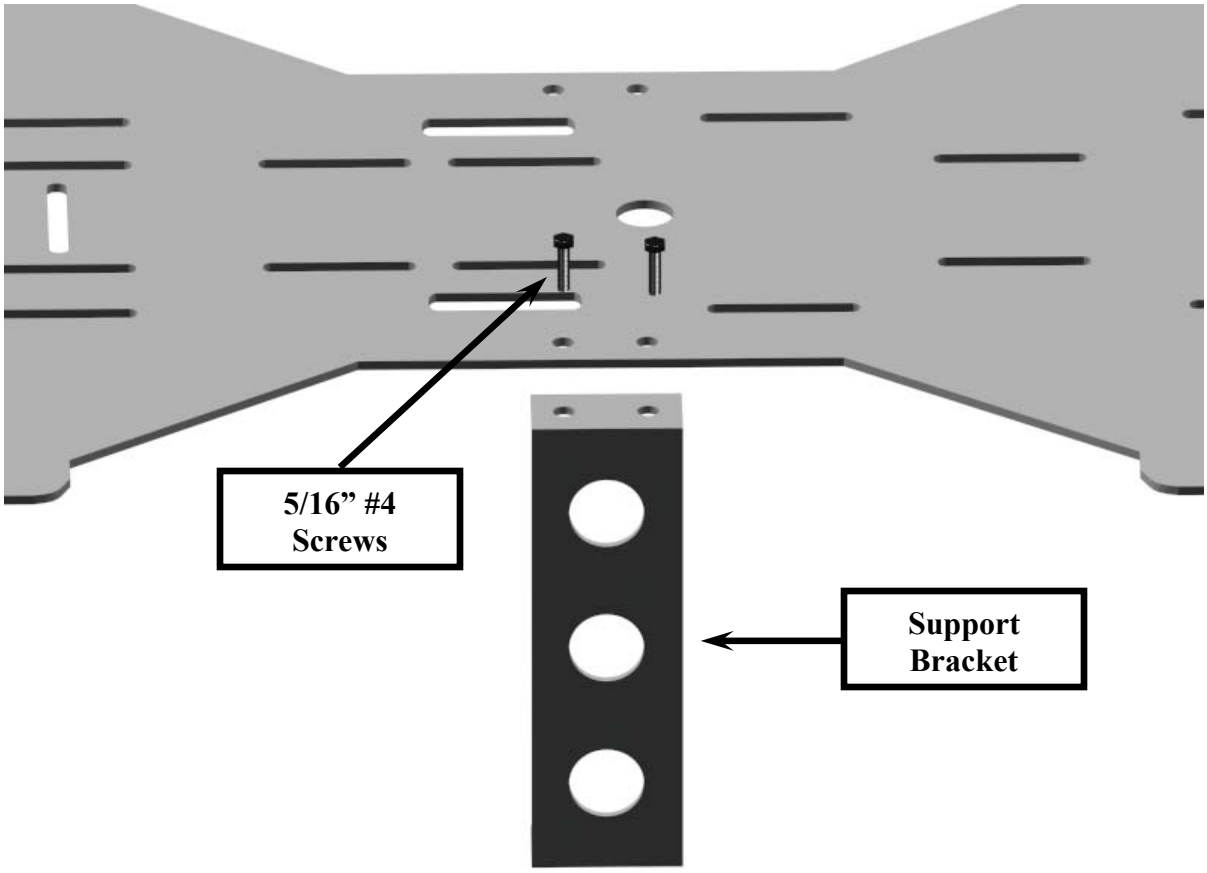



**Figure 30**

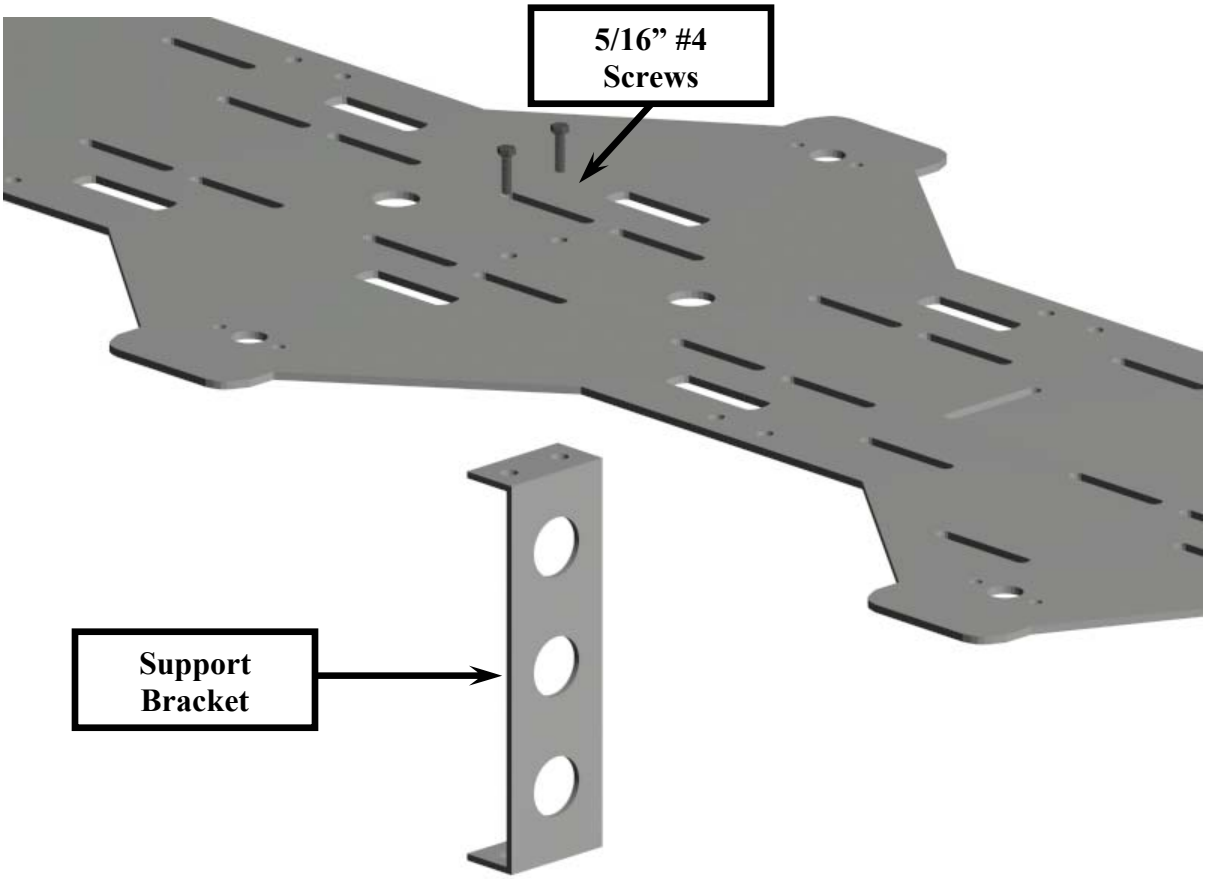



**Figure 30b**

## INSTALLING THE FRONT AND REAR SUPPORTS

5. Using (2) 5/16" #4 screws and washers, install the front and rear support brackets as shown in Figure 31 and Figure 32.

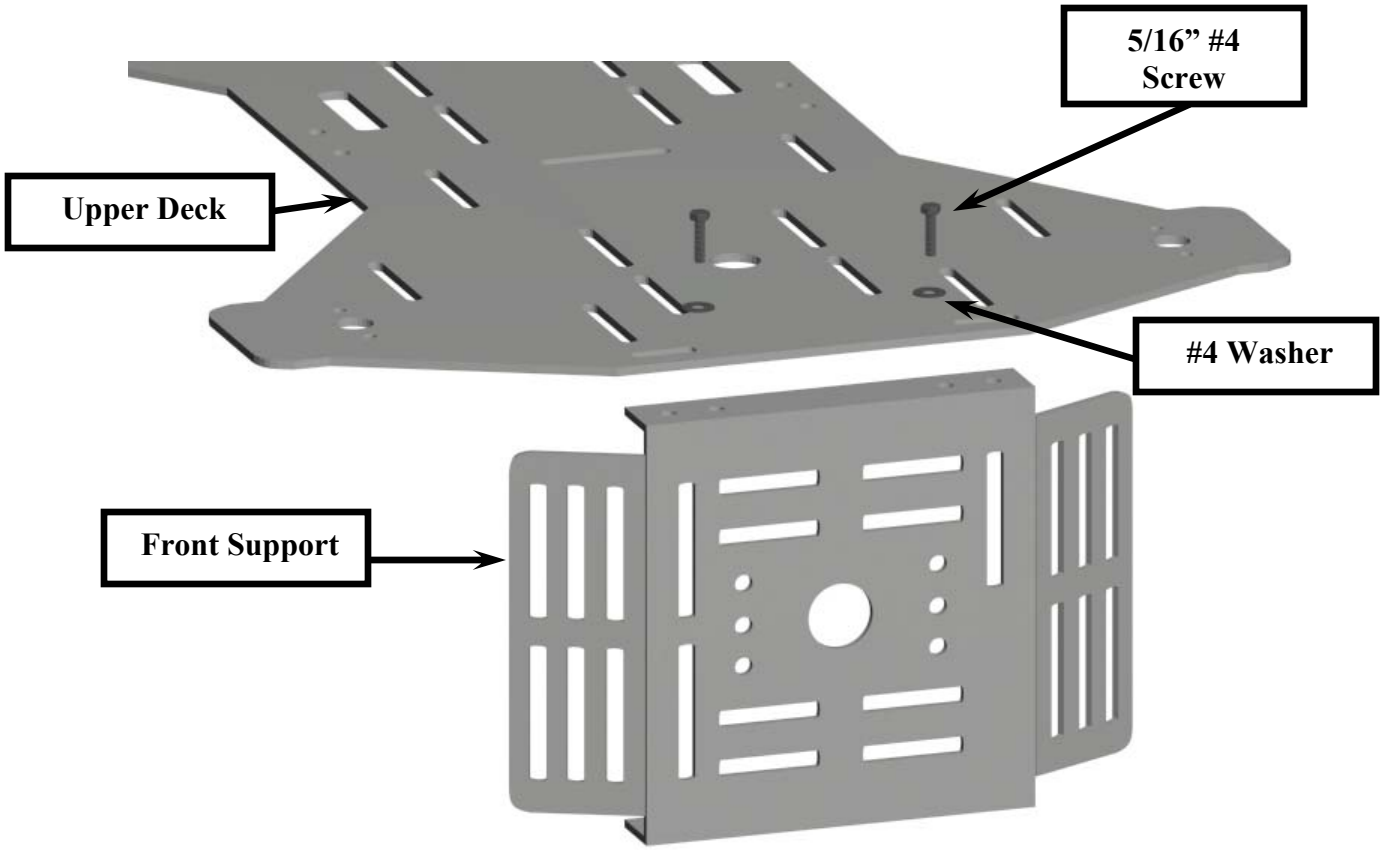



**Figure 31**

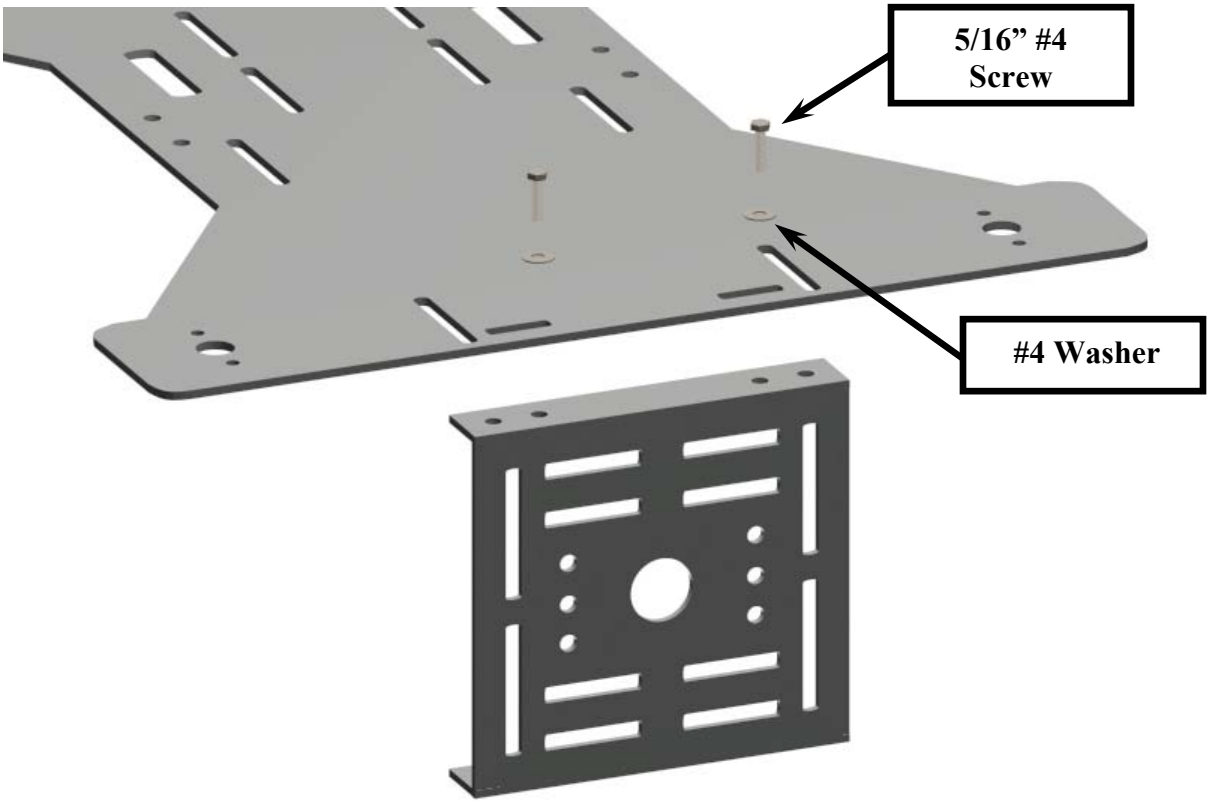



**Figure 32**

## INSTALLING THE LEG ASSEMBLIES TO THE UPPER DECK

1. To ensure that the servo wires do not get in the way of the following steps, secure the servo wires with the cable ties provided in your kit.

| | |
|---|---|
|  | **CAUTION:** Perform the following procedures one leg at a time**.** |

2. Place the assembled upper deck on a clean, flat surface with the front and side braces facing up as shown in Figure 33.

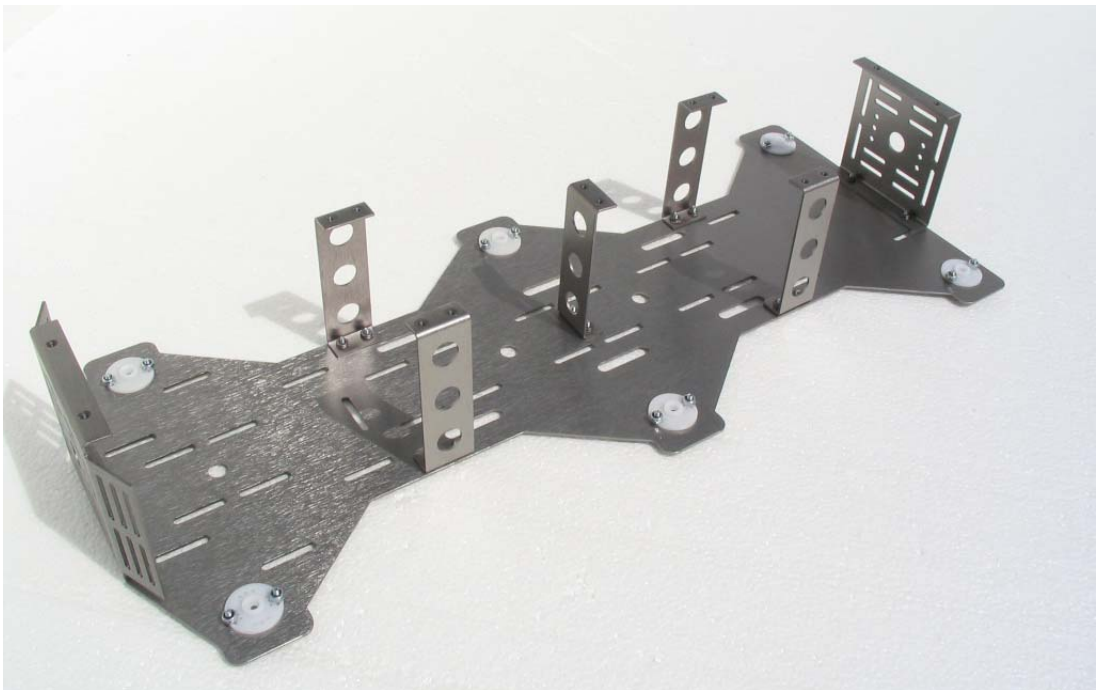


**Figure 33: Upper Deck Assembly**

3. Take a completed leg assembly and insert the legs upper round servo spindle into the upper decks round servo arm as shown in Figure 34.

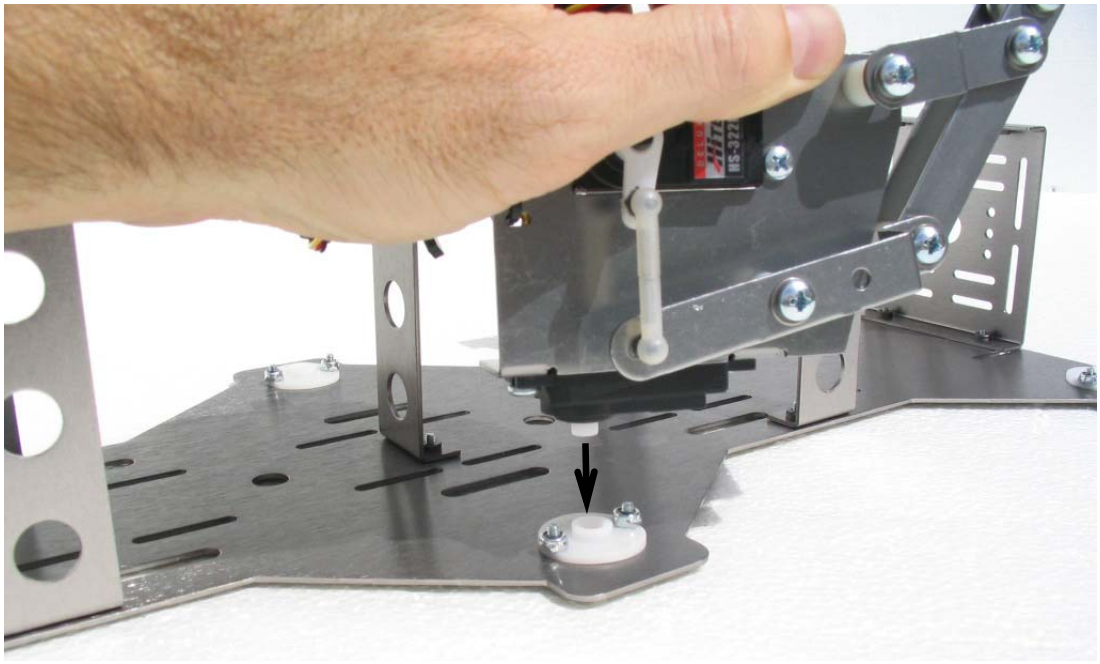


**Figure 34: Installing a Completed Leg Assembly**

| | |
|---|---|
|  | **CAUTION:** Remember to attach the correct leg to the correct side of the upper deck. The back face of the servos should always be facing towards the rear of the upper deck. |

4. Once inserted (do not secure it with the stock servo screw yet), gently swivel and adjust the leg (taking it out of the servo arm rotating the leg clockwise or counterclockwise and then re-attaching it to the servo arm) so that it swings freely from approximately the 10 o'clock position to the 2 o'clock position as illustrated in Figure 35.

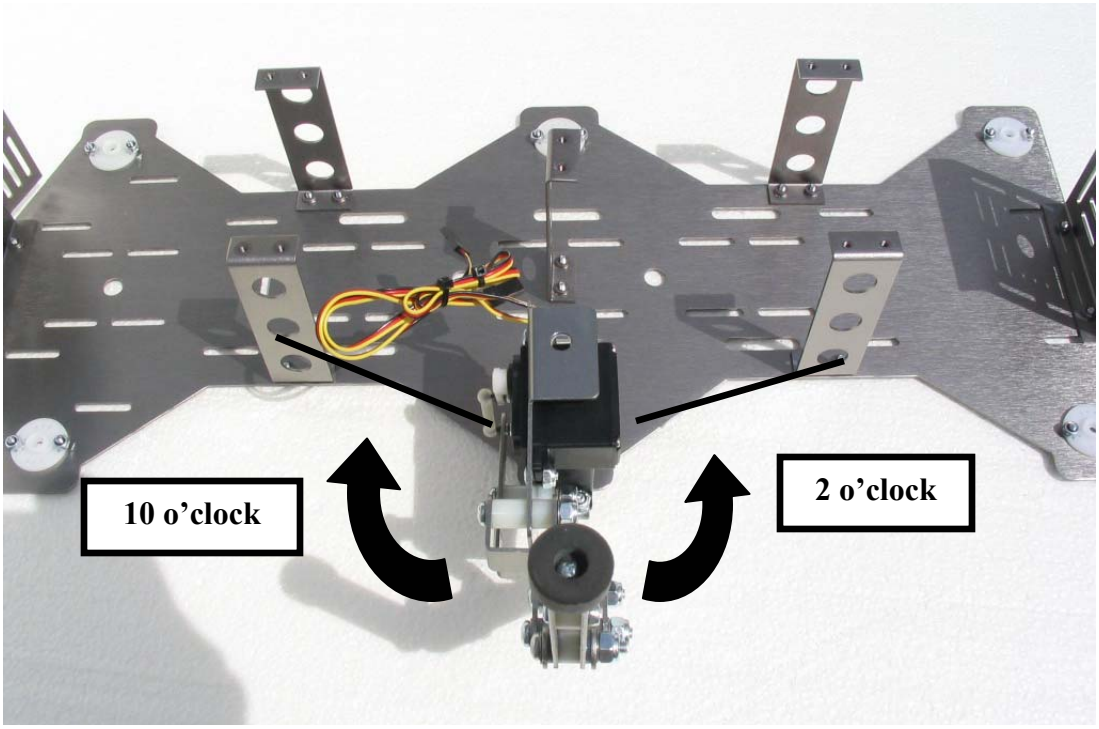



**Figure 35**

5. Once the leg is set correctly, secure the leg using the stock servo screw as shown in Figure 36.


**Figure 36: Securing the Leg Assembly**

6. Repeat steps 3 through 5 for the remaining (5) legs in your kit. Once all of the legs have been installed, your HexCrawler should appear like Figure 37.

7. Place the upper deck assembly to the side of your work area.

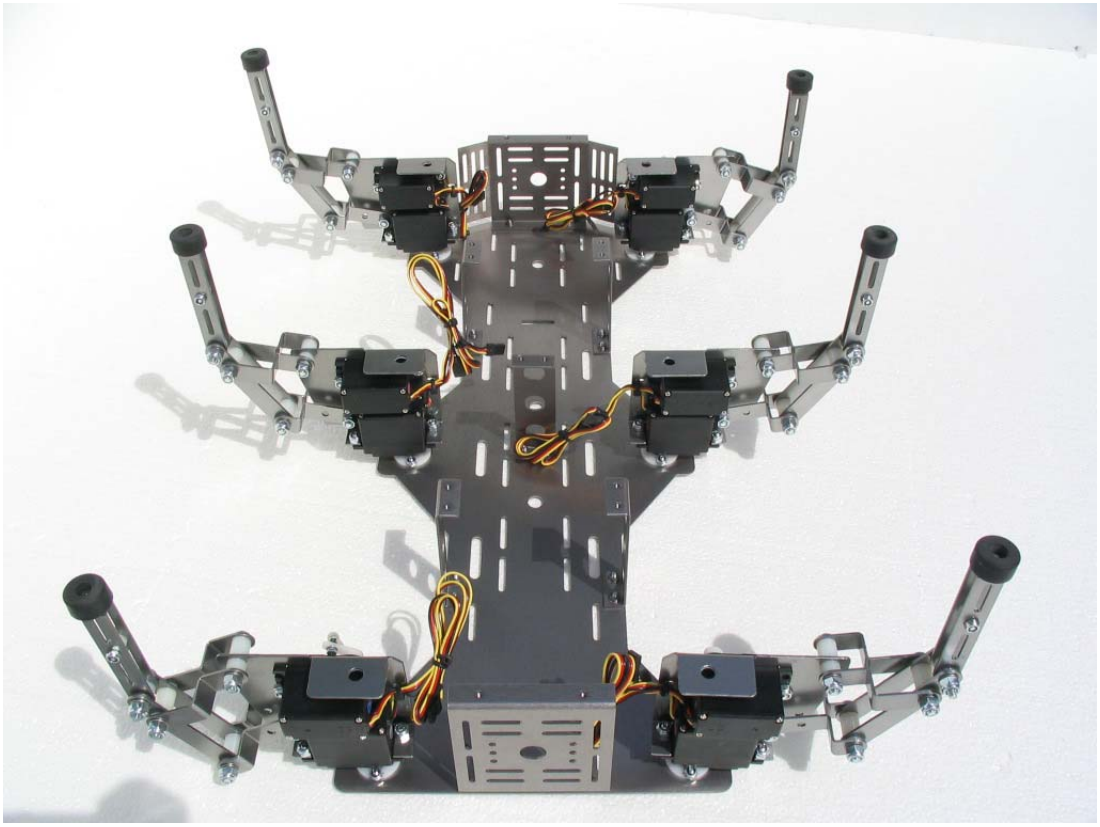


**Figure 37: Completed Upper Deck Assembly**

## PREPARING THE LOWER DECK

1. Apply white grease (or any heavy grease) around the base of all 6 pem studs of the lower deck as shown in Figure 38.

2. Install the SAE ¼ flat washers to each of the pem studs and apply grease to the top of the SAE flat washers as shown in Figure 39.

**Figure 38: Grease Application – Pem Stud**


**Figure 39: Grease Application – SAE Flat Washer**

## ATTACHING THE UPPER DECK ASSEMBLY TO THE LOWER DECK ASSEMBLY

1. Dead-bug assembly is the way to go in the next step. The white grease will hold the washers on the lower deck in place while installing it to the upper deck of the robot. Leaving the robot on its back as shown in figure 40, lower the bottom deck onto the bottom of the HexCrawler being sure to align the pen studs with the leg pivot holes of the servo holders.

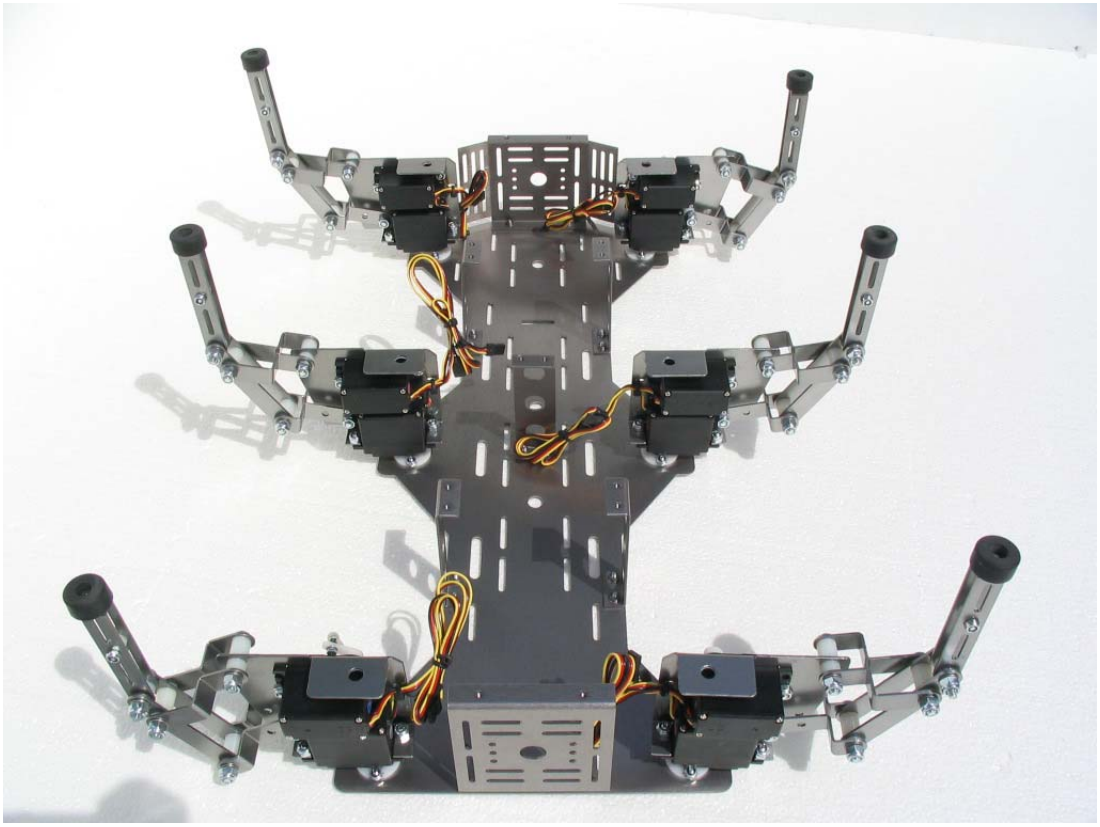


**Figure 40**

2. Once **all** of the pem studs are aligned and inserted into the leg pivot holes, secure the lower deck to the upper deck with (10) #4 - 5/16" screws and (4) #4 washers (where applicable) as shown in Figures 41, 42 and 43.

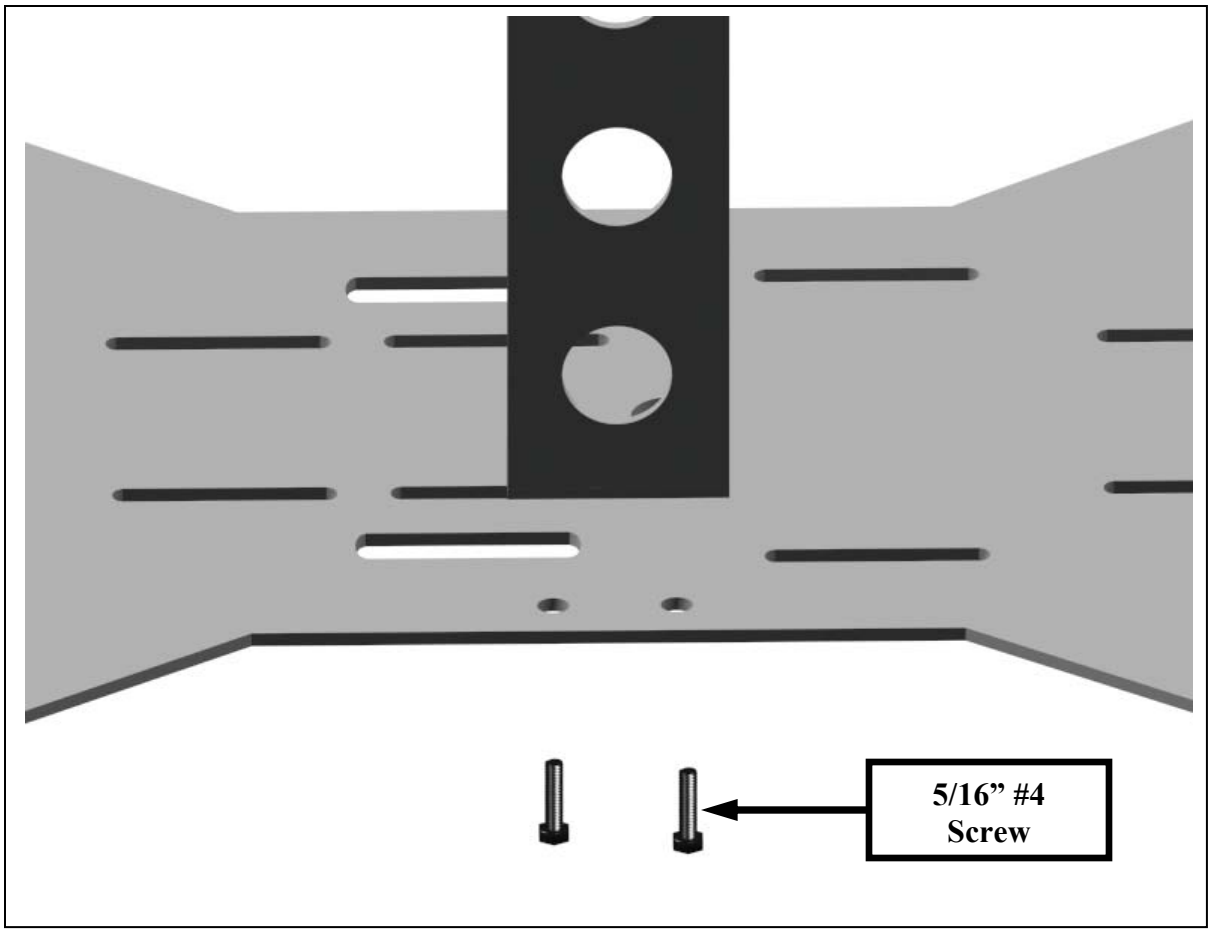



**Figure 41: Side Support Assembly**

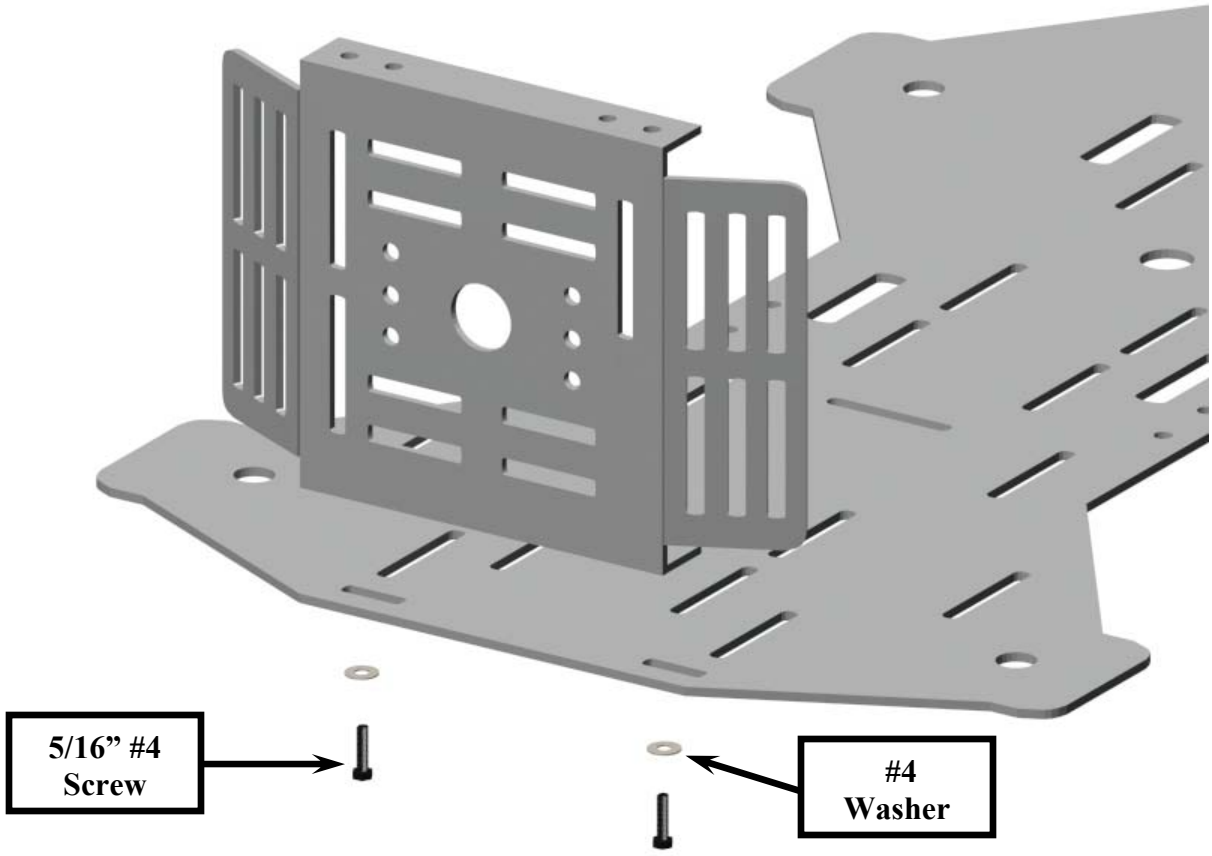



**Figure 42: Front Support Assembly**

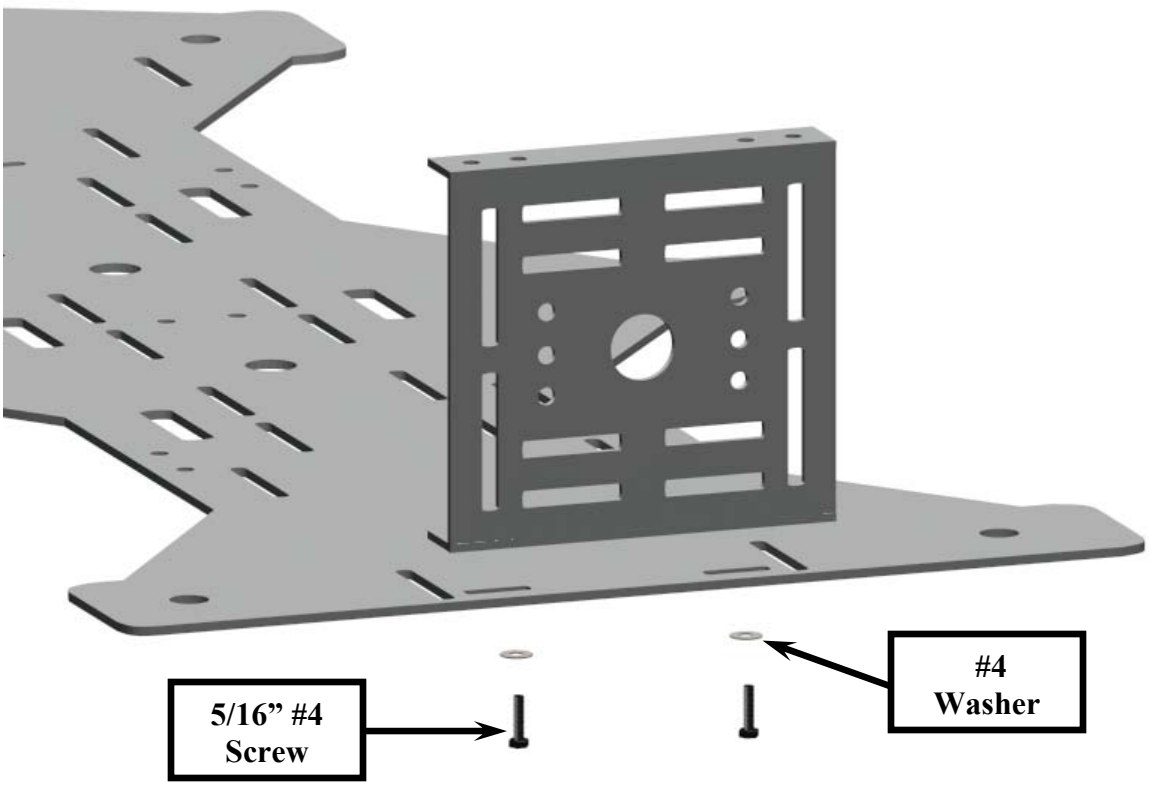



**Figure 43: Rear Support Assembly**

5. Carefully place the HexCrawler back onto the elevated platform in the standing position.

## INSTALLING THE DOG BONES AND SERVO ARMS

1. With a 1/8” drill bit, drill out the second hole from the center hole of the servo arm as shown in Figure 44. It does not matter which of the (2) arms are drilled. To secure the servo arm while its being drilled, place the arm on an available servo’s spindle.


**Figure 44**

2. Attach the ball link with the #2 washer, lock washer and lock nut (Figure 45). The ball should be orientated towards the flat side of the servo arm.

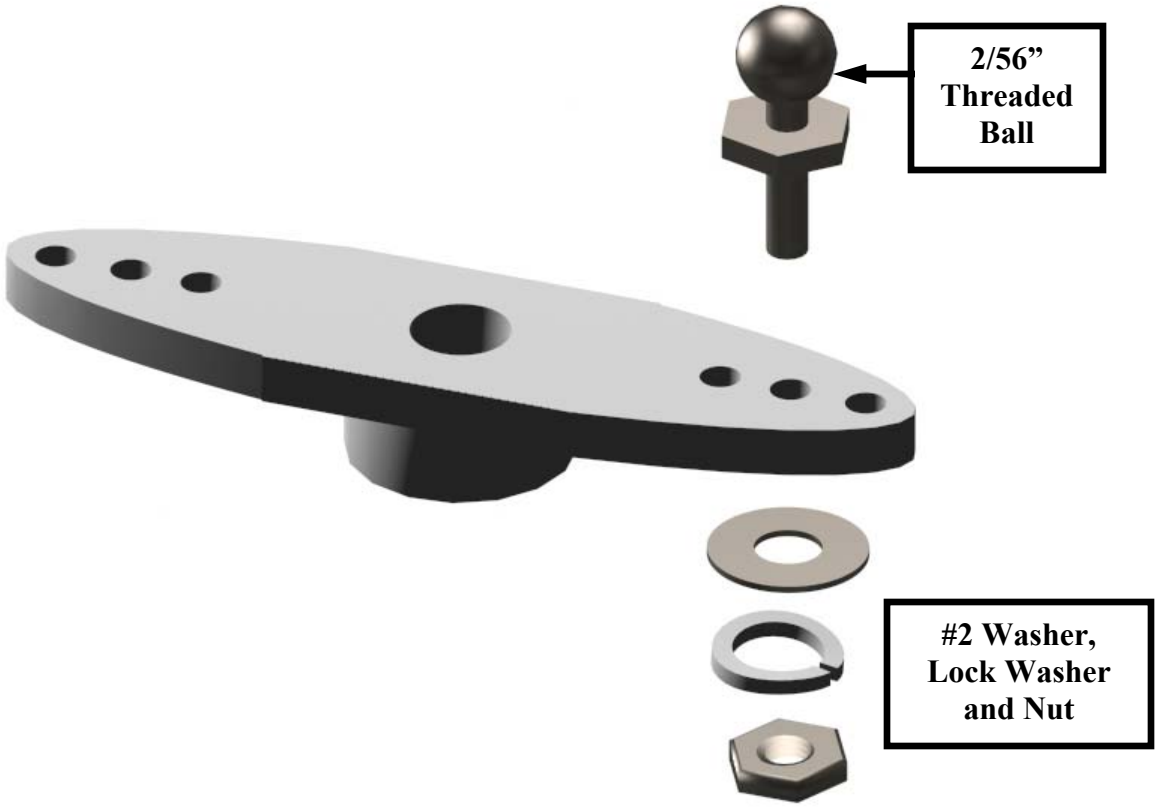



**Figure 45**

3. With the 2/56” threaded ball oriented upwards, install the servo arm to the servo spindle on the lower servo of the leg assembly and secure the assembly with the stock servo screw (see Figure 46).

**Note:** Cut off the remaining arm on the servo horn, which is not being used before the threaded ball is installed. Failure to remove the remaining servo arm will result with the servo arm hitting the servo spindle during the HexCrawlers walking sequence.


**Figure 46**

4. Join the 2 dog bones by threading the 2/56" threaded rod into each end of the dog-bones. Install the threaded rod completely into 1 dog-bone before threading the other (Figure 47). Repeat this step for all of the legs on your HexCrawler. This is often easiest done with a small vise. Don't grip the dog bones or the threaded rod too hard or you could damage the parts.

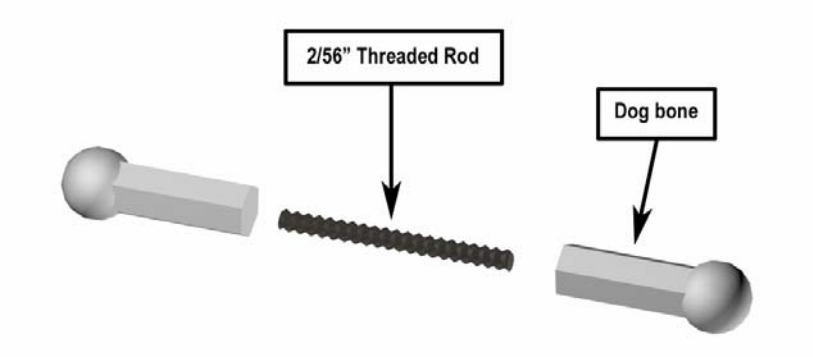



**Figure 47: Dog Bone Assembly**

| | |
|---|---|
|  | **Tip**: If turning the threaded rod onto the dog bone is difficult by hand, place a small drop of oil or apply a small amount of grease to the threaded rod before installing it into the dog bone. |

**Figure 48: Completed Dog Bone Assembly**

| | |
|---|---|
|  | **CAUTION**: If your dog bones do not thread together correctly, don't use glue to hold them in place. You have two choices for fixing this problem:<br>• Contact Parallax for replacement parts via info@parallax.com.<br>• Go to a local hobby shop and get Du-Bro #188 ball link sockets for replacement sockets and standard 2/56 threaded rod. |

5. Snap on the assembled dog-bone to the 2 ball joints on the leg assembly. Repeat steps 4 and 5 for each leg (see Figures 49 and Figure 50).

| | |
|---|---|
|  | **Warning:** Ensure that you support the back of the main leg actuator when snapping on the dog-bone to the ball joint or the leg actuator may bend with excessive force. |

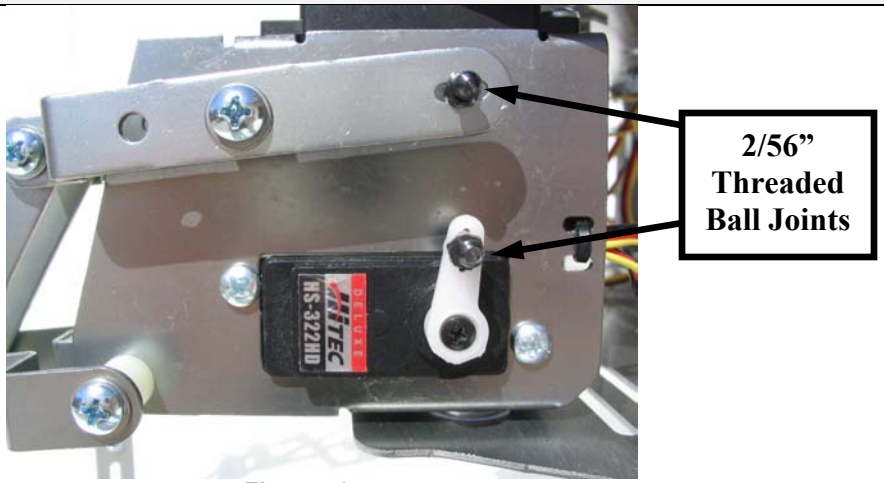



**Figure 49**

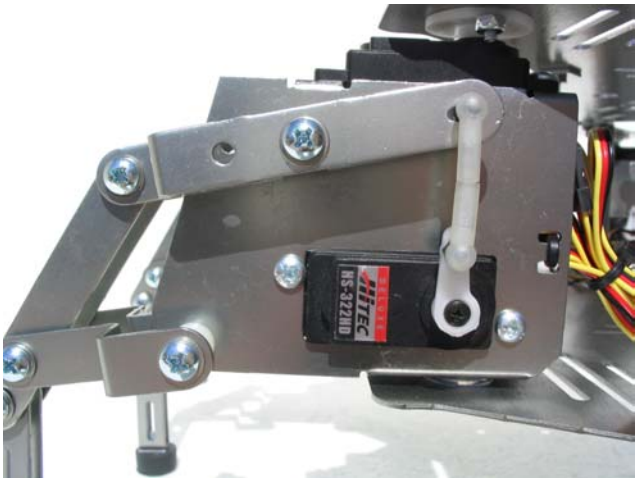


**Figure 50**

6. **This is one of the most important steps in this manual.** Facing the front of the HexCrawler so the servo horns are showing, gently turn and re-install as necessary the servo control horn (keep the screw off for the moment) so that it swings from 11 to 7 o'clock in the **clockwise** direction for the legs on the HexCrawler's **right side** and from 1 to 7 o'clock in the **counterclockwise** direction for the legs on the **left side** of the HexCrawler (see Figures 51 and 52 below).

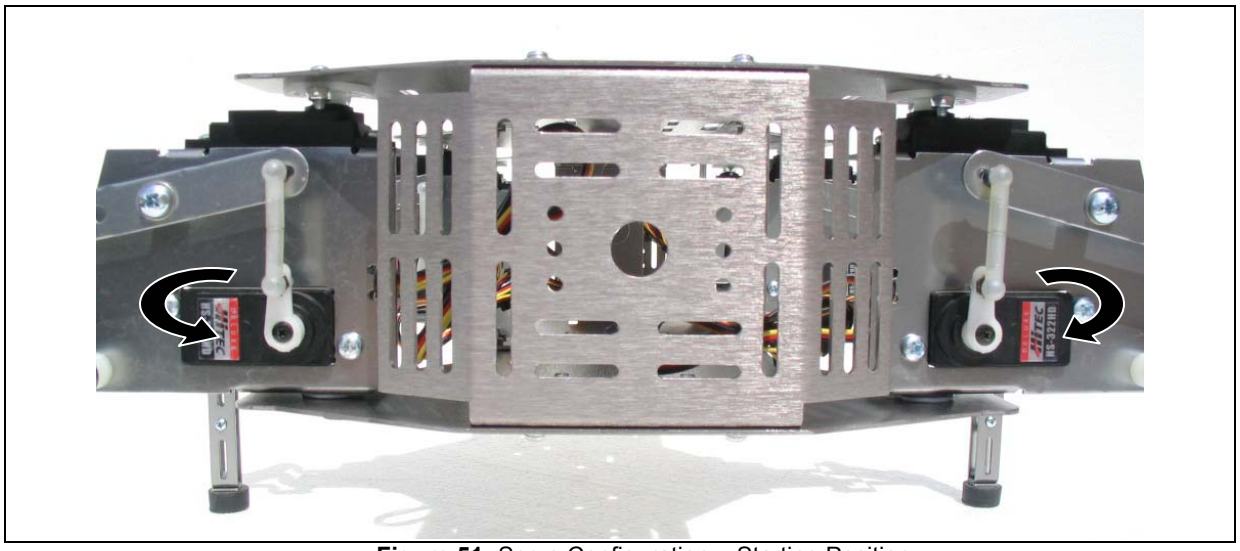


**Figure 51:** Servo Configuration – Starting Position

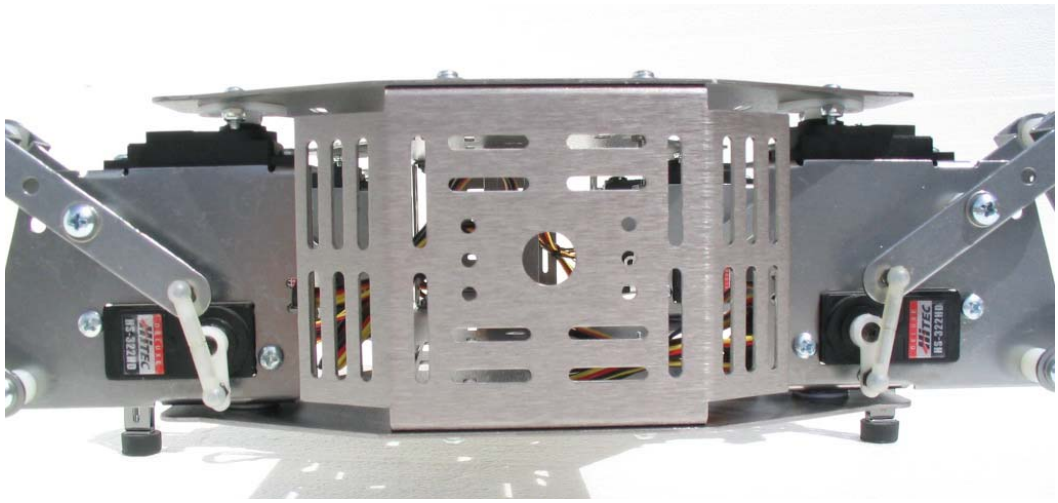


**Figure 52:** Servo Configuration – Ending Position

Repeat steps 1 – 6 for all of the legs on your HexCrawler**.**

## "TUNING" THE LEGS OF YOUR HEXCRAWLER

1. Using a 11/32" wrench or socket along with a Phillips screwdriver, tighten all of the lock nuts on the 1.25" #8 screws just tight enough so they can be turned by hand. (screws 1 – 4 only, see Figure 54)

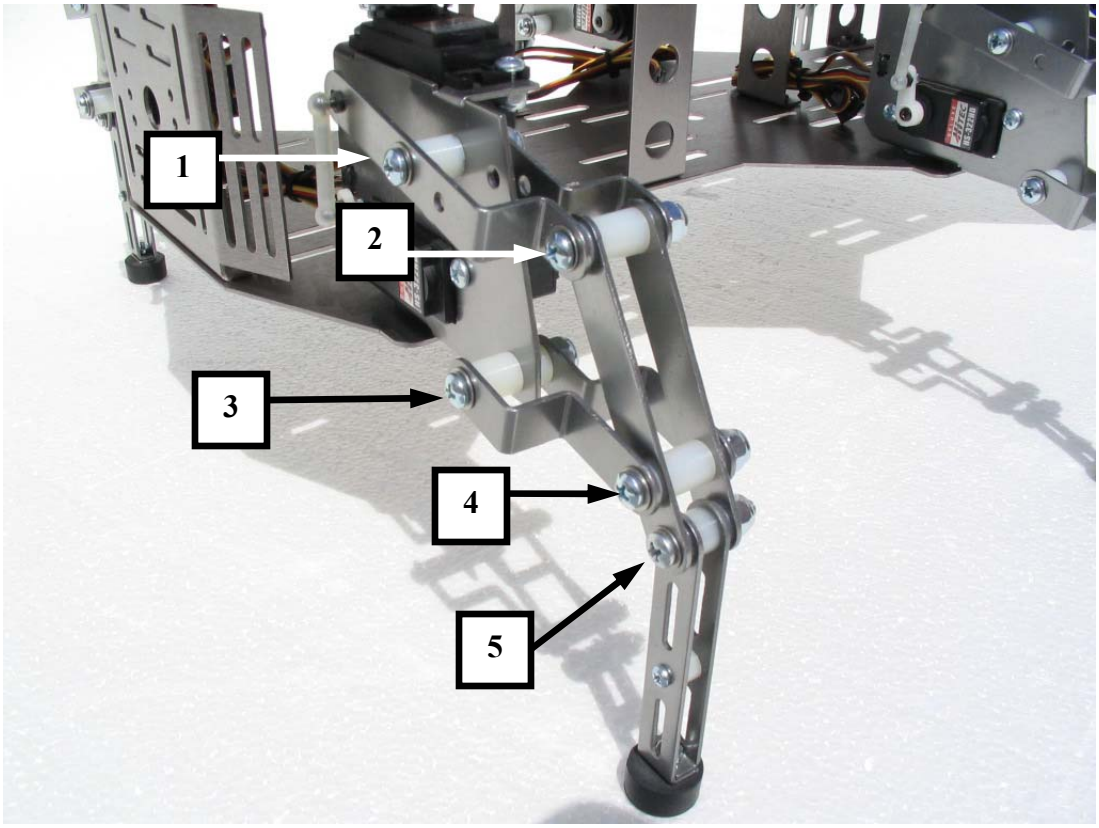



**Figure 53**

2. Ensure the lower leg is as vertical as possible (for best accuracy, use a level) and then completely tighten the 1" #8 screw with the locknut (box 5 in Figure 53).

3. Repeat the steps above for all of the legs on the HexCrawler.

4. After tightening each leg as per the steps outlined above, perform the following leg screw check:

- **Tight Screws** - Move the vertical lift servo so that the leg moves up and down and ensure that the leg moves freely and is not difficult to move or appears stiff. If the leg is difficult to move or appears stiff, one or more screws are too tight. **Hand** turn each screw until the tight screw(s) are found and loosen them just enough so that they can be turned by hand.

- **Loose Screws** - Move the vertical lift servo so that the leg moves up and down and ensure that the leg moves freely and that there is not too much horizontal play or "rattle" in the leg. If the leg appears loose, hand check each screw and tighten the screw just enough so that it can be turned by hand.

## REFERENCE PICTURES

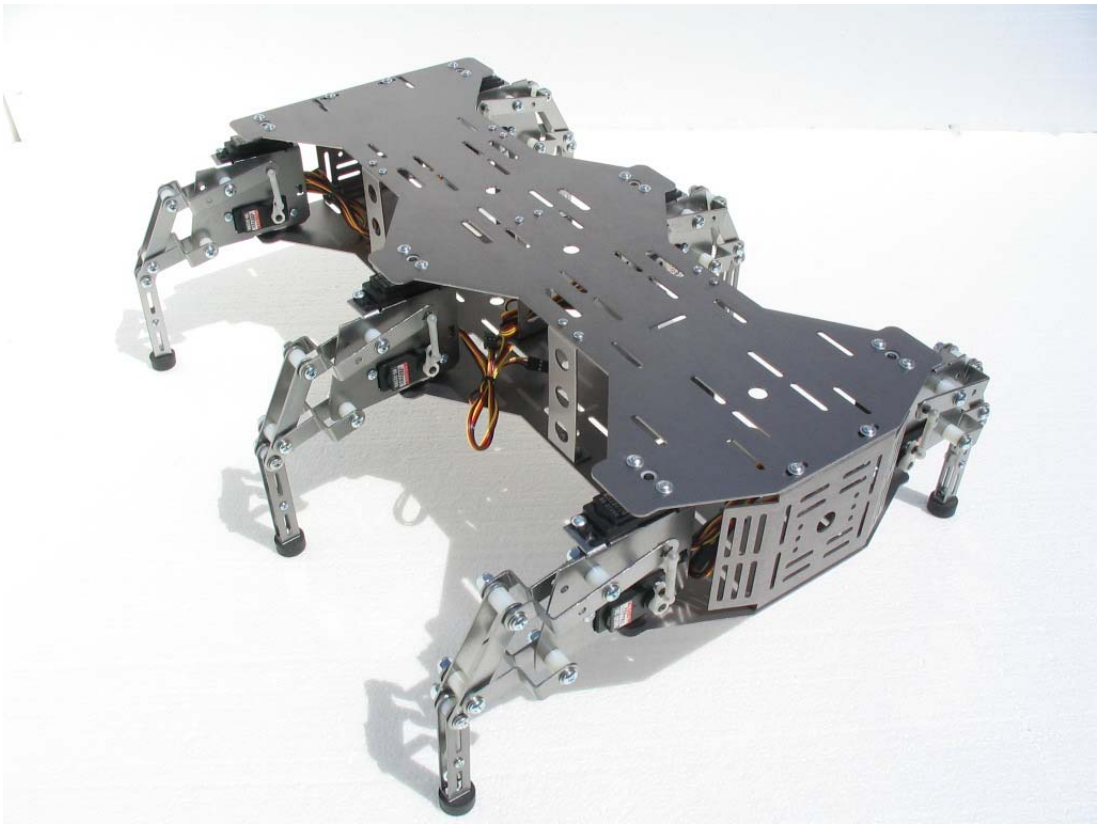


**Figure 54: HexCrawler – Front View- Standing**

**Figure 55:** Front View - Squatting

**Figure 56:** HexCrawler - Rear View

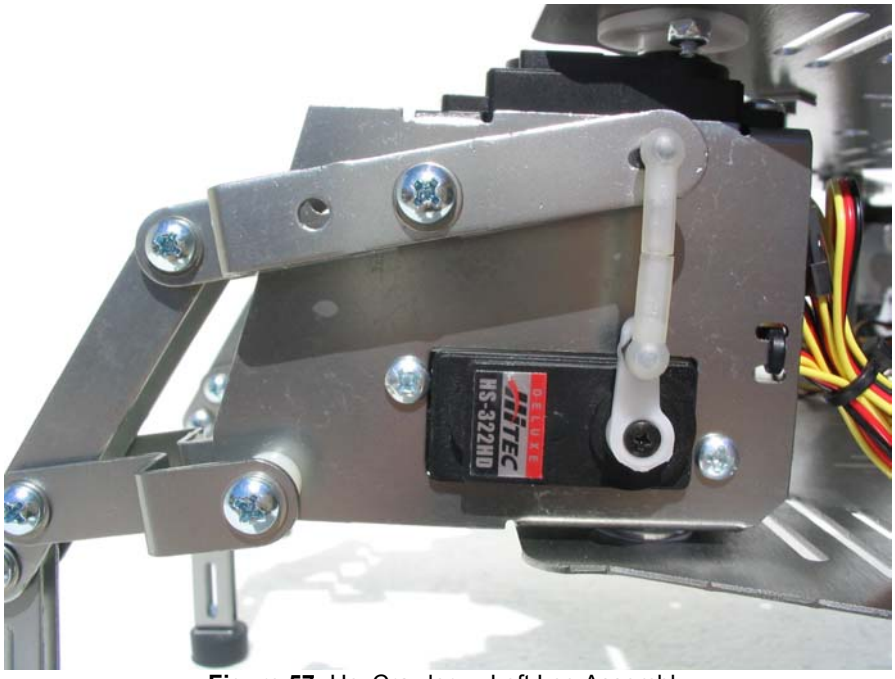



**Figure 57:** HexCrawler - Left Leg Assembly

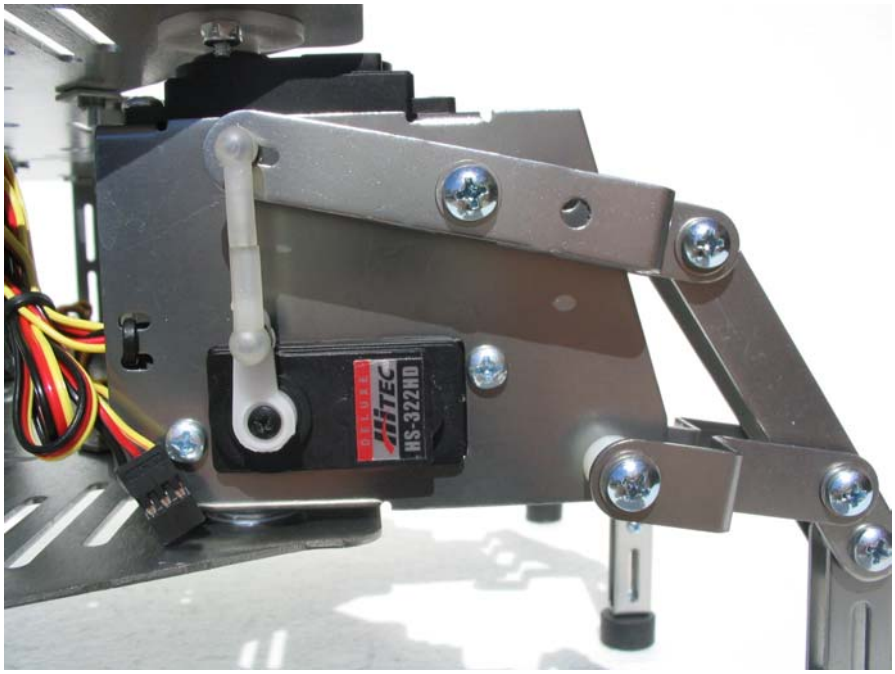



**Figure 58:** HexCrawler - Right Leg Assembly

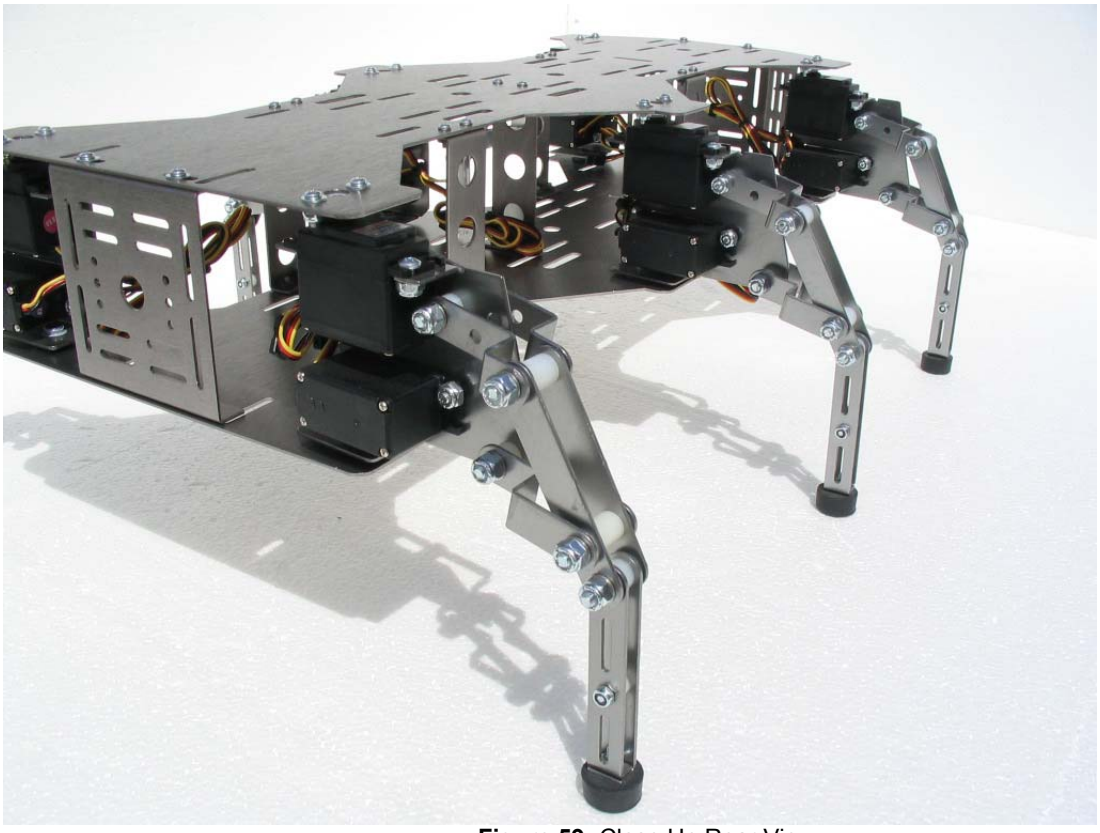


**Figure 59:** Close Up Rear View

## BOARD OF EDUCATION AND PARALLAX SERVO CONTROLLER PLACEMENT

Use the following pictures for the placement of the Parallax Board of Education carrier board (BOE) and Parallax Servo Controller (PSC). Between the 2 decks of the HexCrawler, we have provided 4 different places to secure the BOE board and 8 places to secure the PSC board onto the HexCrawler. The placement choice is strictly up to you. Use #4 - ½" screws, #4 - 3/16" nylon spacers and #4 nuts provided in your HexCrawler hardware kit. In the example that follows, the BOE will be mounted onto the rear slots of the top deck of the HexCrawler and the PSC will be mounted to the middle of the top deck as shown in figures 61 and 62.

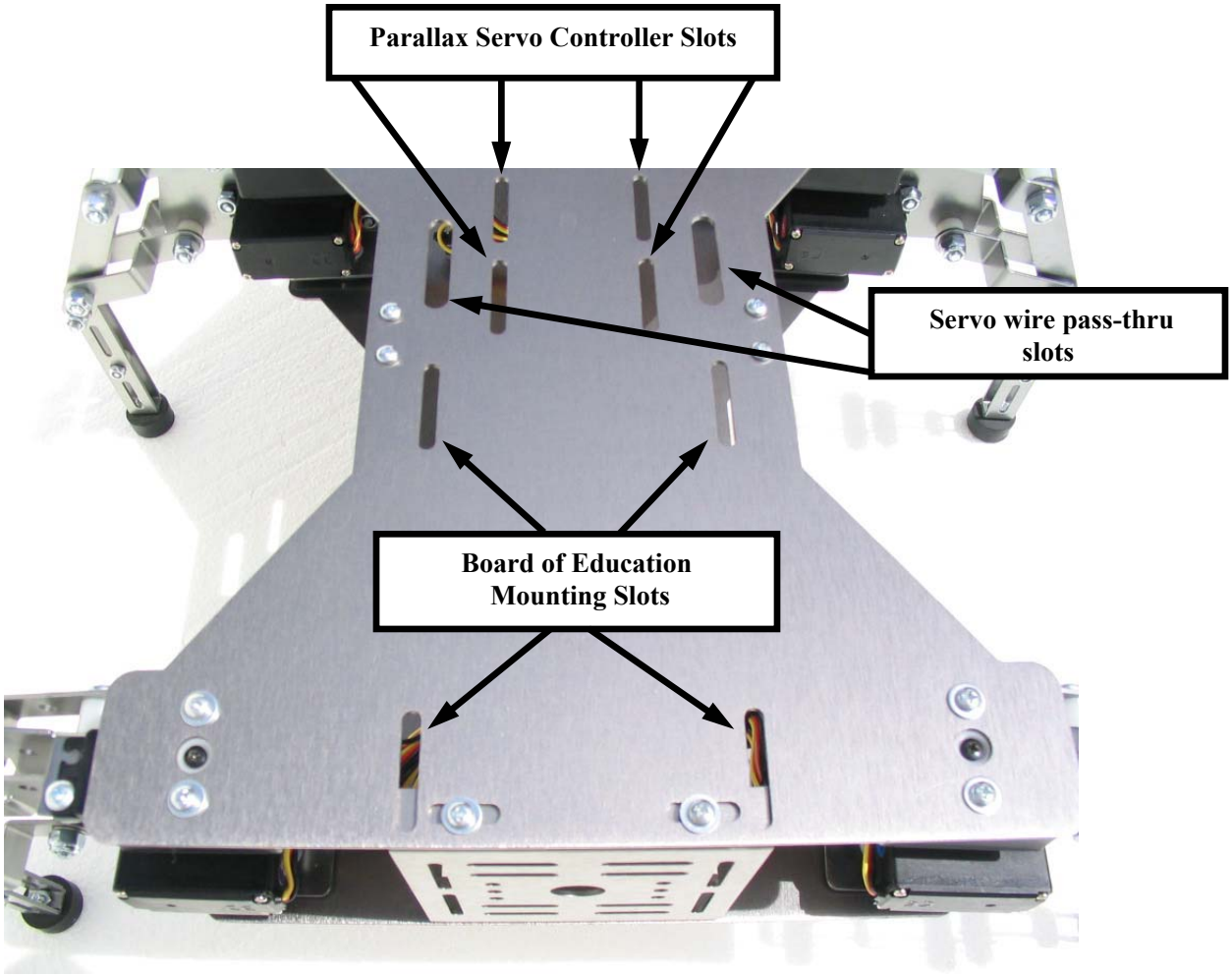



**Figure 60**

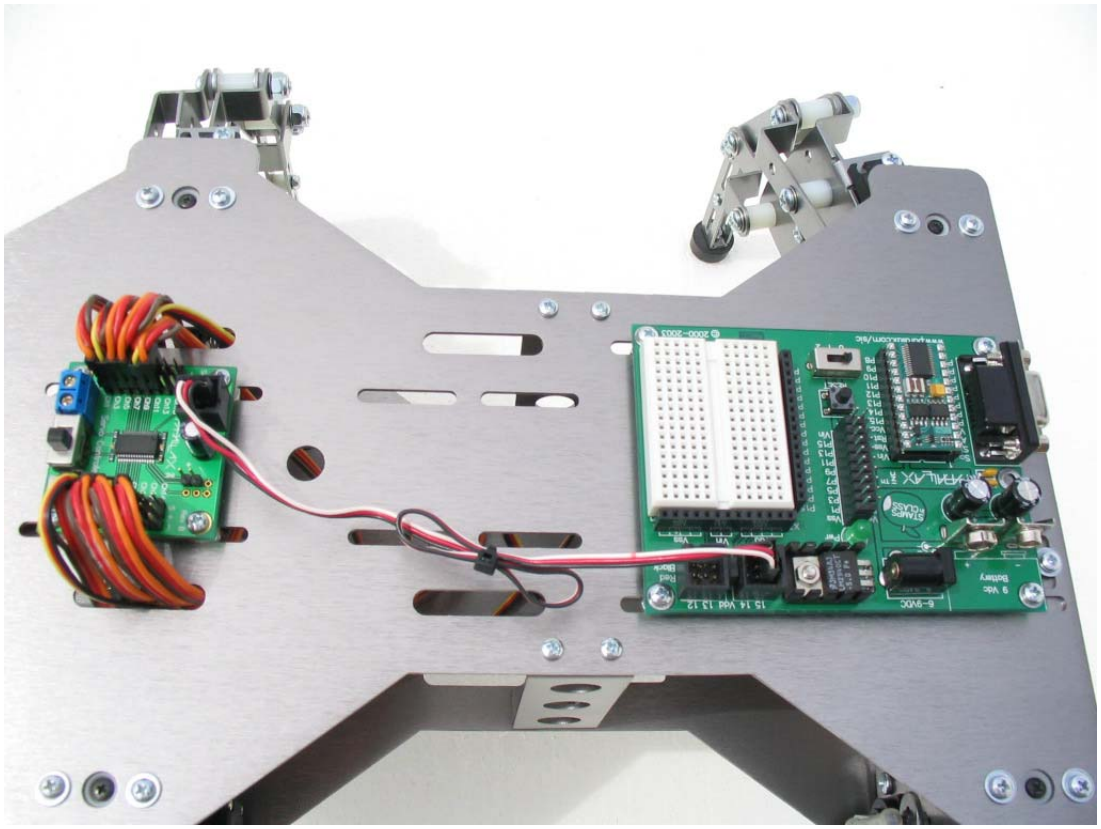


**Figure 61**

**Note:** The servo and BOE wiring shown above will be covered in the next chapter.

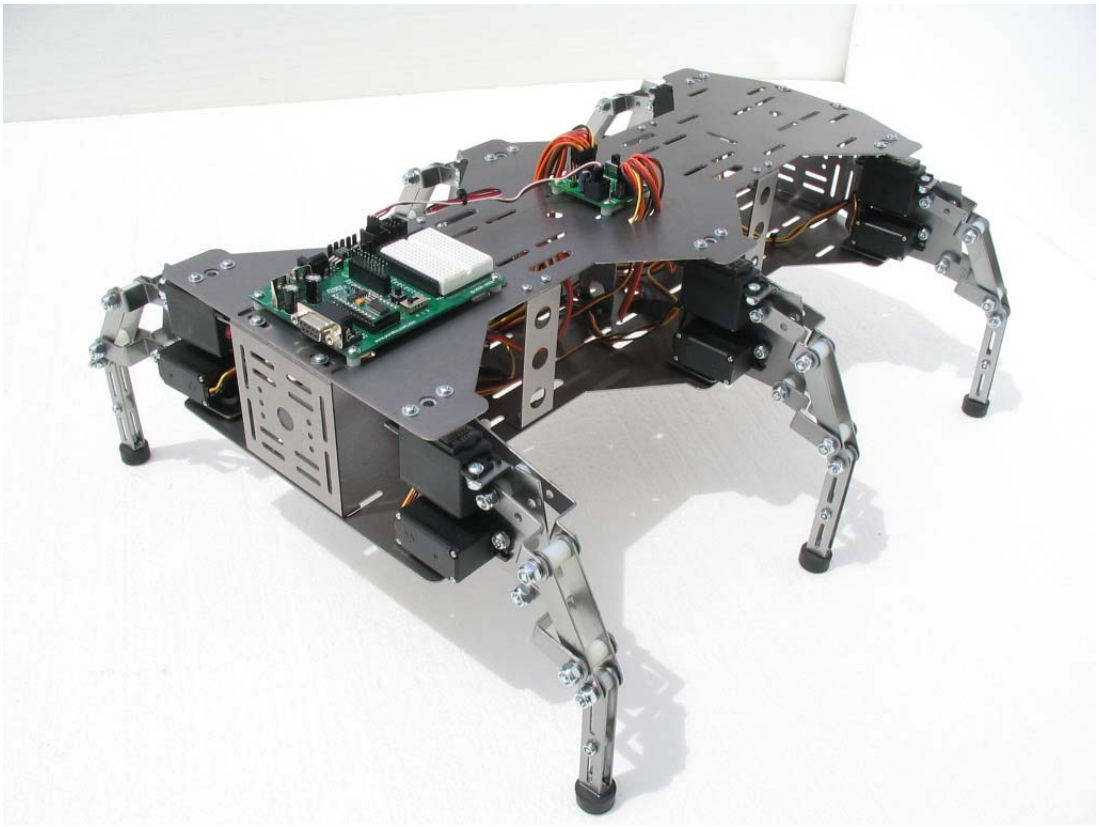


**Figure 62**

# Chapter #4: Wiring and Tuning the HexCrawler

This section describes how to wire circuit used in the HexCrawler example programs in the next chapter.

## PARTS REQUIRED

(1) BASIC Stamp 2 IC module on a Board of Education carrier board
(1) Parallax Servo Controller (PSC)
(12) Hitec Servos (already installed on the HexCrawler)
(1) Seven-segment green LED (Fairchild MAN6480)
(7) 1 kΩ resistors
(2) Pushbuttons
(2) 10 kΩ resistors
(2) 220 Ω resistors
(1) 7.2 V battery
(1) 9 V transistor battery (if you intend on powering the BASIC Stamp module separately)
(misc) jumper wires

## CONNECT THE PARALLAX SERVO CONTROLLER

Connect the HexCrawler's servos to the Parallax Servo Controller a shown. Connect the Parallax Servo Controller to the Board of Education carrier board.

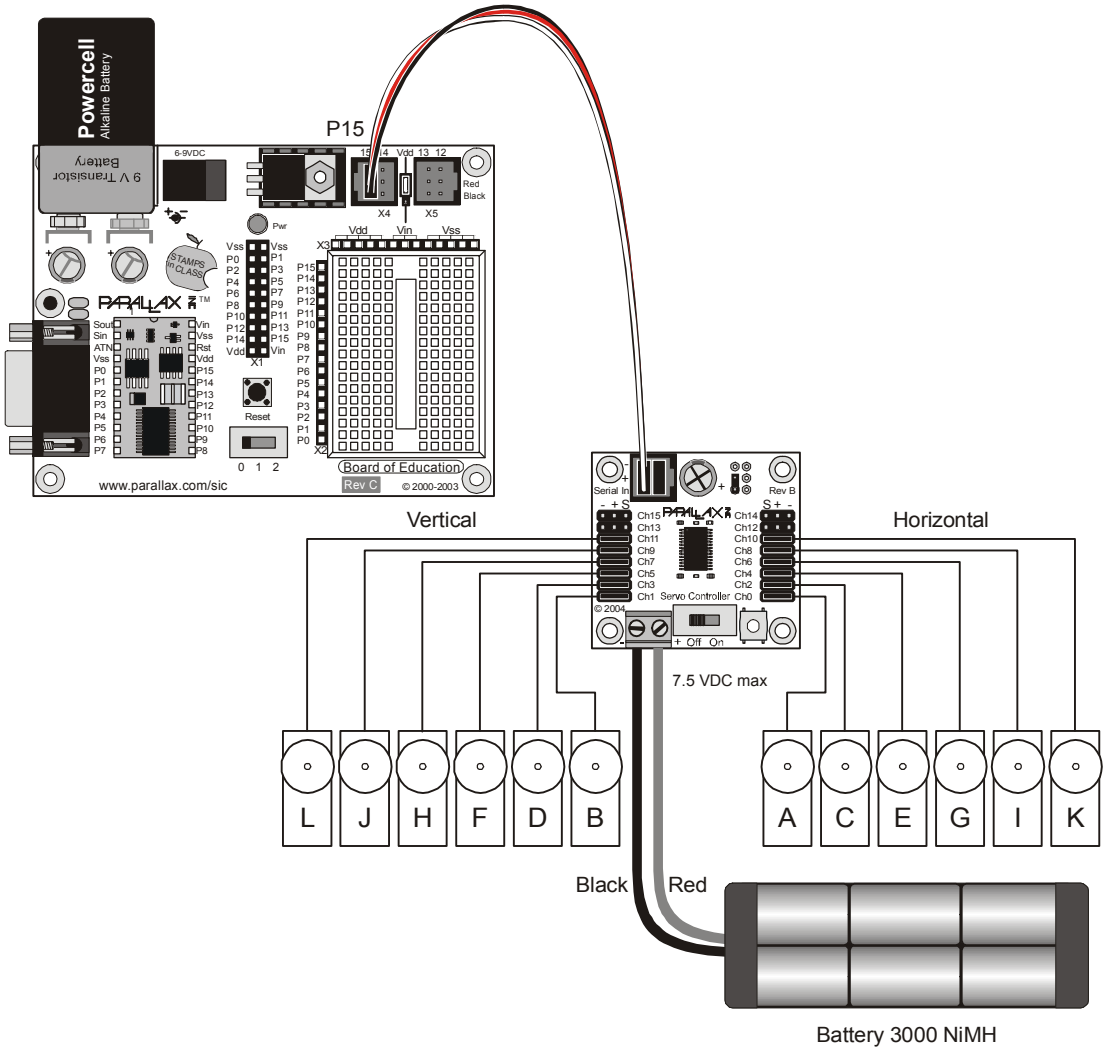
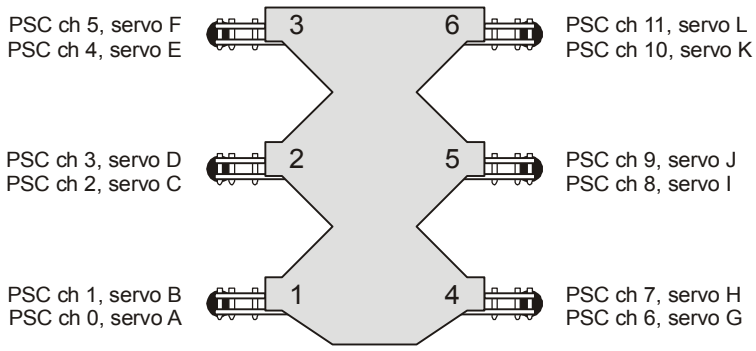



**Figure 63**




Servos moving legs vertically connect to odd numbered PSC ports.
Servos moving legs horizontally connect to even numbered PSC ports.

**Figure 64**

## PUSHBUTTON AND LED CIRCUIT

The HexCrawler's example program is run using a two-pushbutton, seven-segment LED display system feedback. One pushbutton starts/stops the gait and the other selects the gait to be used.

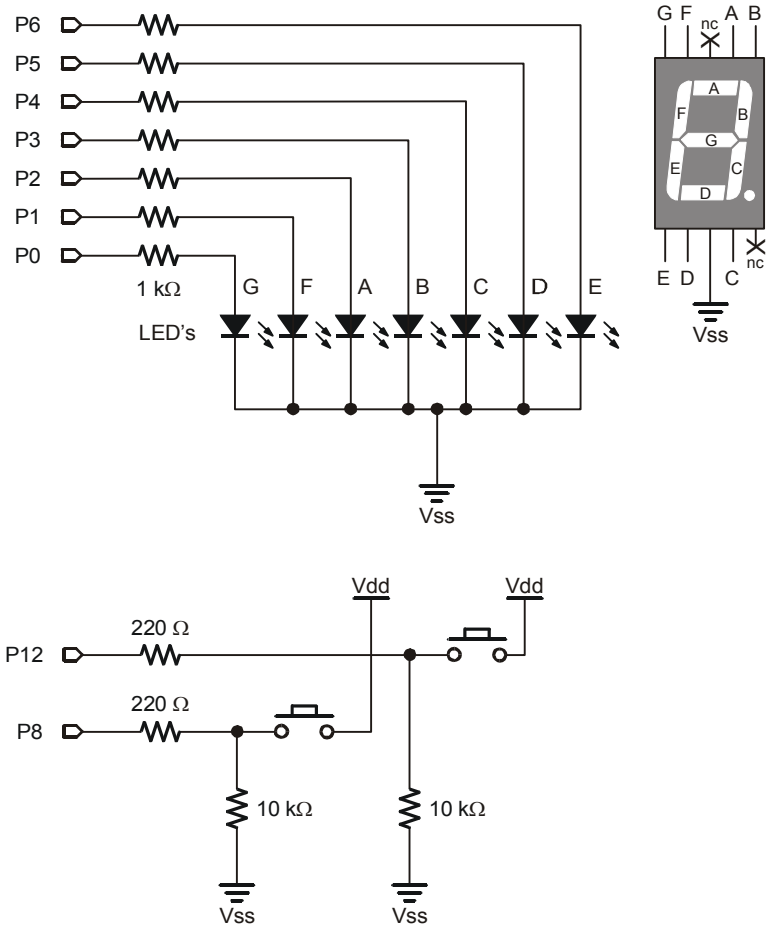



**Figure 65: Pushbutton Control System Schematic**

### POWER SUPPLY CONNECTIONS

This is the trickiest part of finishing the circuit.

- Connect the Parallax Servo Controller power supply directly to the 7.2V battery.
- Connect the Board of Education carrier board to the 9V battery.

## NETWORKING ADDITIONAL PARALLAX SERVO CONTROLLERS

One Parallax Servo Controller supports 16 servos. If you add many servos to your HexCrawler for the 3 degree of freedom leg (3DOF) upgrade or a robotic arm you would simply network another Parallax Servo Controller as shown below.

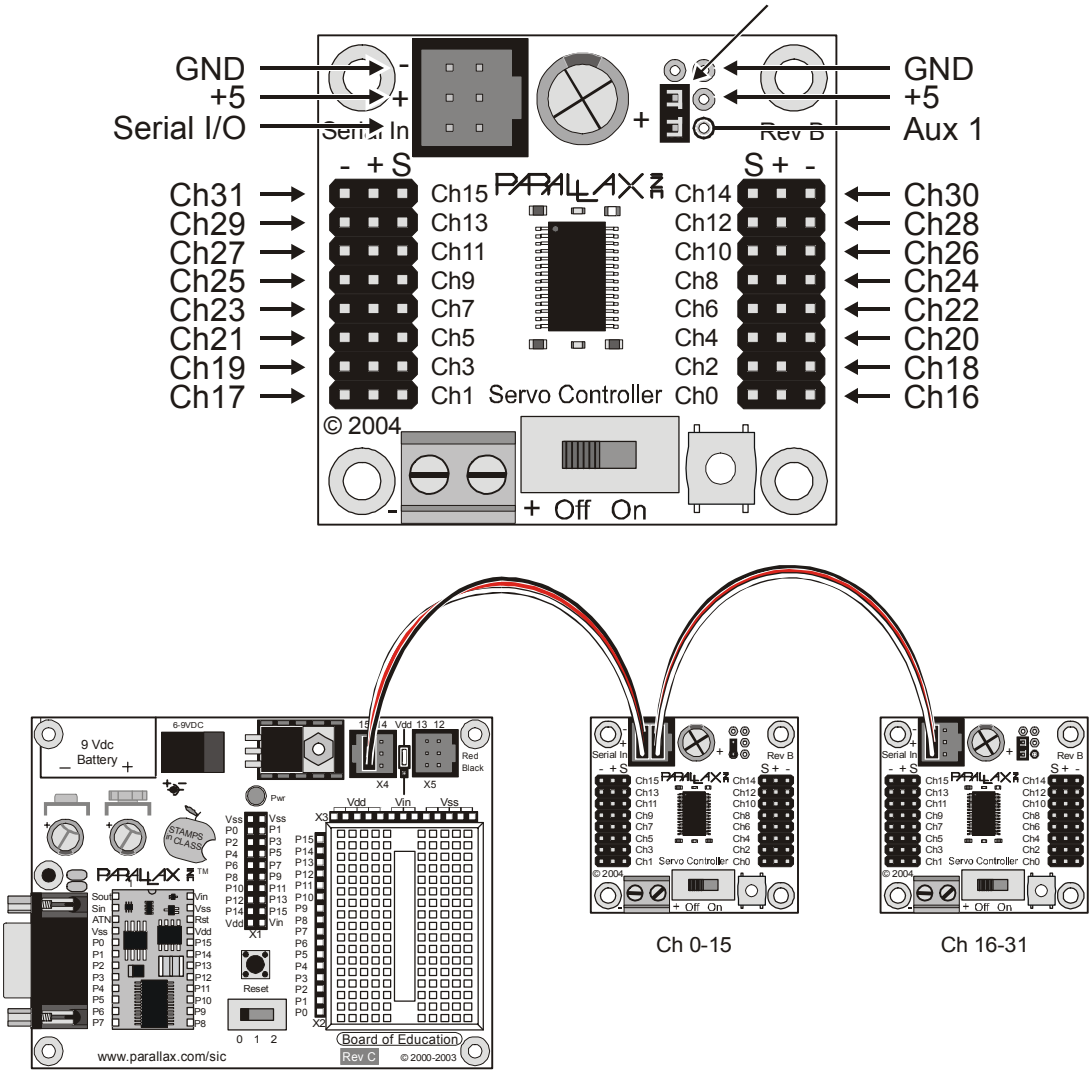



**Figure 66**

# Chapter #5: Programming the HexCrawler

Prior to running the HexCrawler we will check the individual circuit systems (the pushbuttons, the LED) and then configure the servos for running. All of the programs used in this chapter are available for download from the HexCrawler page on the Parallax web site. If you need help understanding how to setup the BASIC Stamp Windows editor and download programs see the *What's a Microcontroller?* Version 2.0 text.



## TESTING THE PUSHBUTTON CONTROLS

Run the following code to check the pushbuttons:

```
' =============================================================================
'
'   File....... HexCrawler Pushbutton Test.BS2
'   Purpose.... Simple checkout of pushbutton circuits
'   Author..... Parallax
'   E-mail..... support@parallax.com
'   Started....
'   Updated.... 11 AUG 2003
'
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' =============================================================================


' -----[ Program Description ]---------------------------------------------

' Uses editor DEBUG screen to check button inputs.


' -----[ Revision History ]------------------------------------------------


' -----[ I/O Definitions ]-------------------------------------------------

Button1         PIN     8
Button2         PIN     12


' -----[ Constants ]-------------------------------------------------------


' -----[ Variables ]-------------------------------------------------------

btnVal          VAR     Bit                     ' state of a button


' -----[ EEPROM Data ]-----------------------------------------------------
```

```
' -----[ Initialization ]--------------------------------------------------

Setup:
  DEBUG CLS, "Button Test"


' -----[ Program Code ]-----------------------------------------------------

Main:
  DEBUG CRSRXY, 0, 2
  btnVal = Button1                              ' get state of button 1
  DEBUG "Button on P8.... ", BIN1 btnVal        ' display it
  GOSUB Show_State
  btnVal = Button2                              ' get state of button 2
  DEBUG "Button on P12... ", BIN1 btnVal        ' display it
  GOSUB Show State
  GOTO Main                                     ' do it again


' -----[ Subroutines ]------------------------------------------------------

Show State:
  IF (btnVal = 1) THEN
    DEBUG " (Pressed)", CLREOL, CR
  ELSE
    DEBUG " (Not Pressed)", CLREOL, CR
  ENDIF
  RETURN
```

If the circuit is wired correctly you will be able to press either pushbutton and see a “1” appear in the Debug Terminal as shown in Figure 60. If you receive no response from the pushbuttons carefully check the wiring to see that you are connected to the correct BASIC Stamp I/O pins. Once it is working properly proceed to the next section to test the seven-segment LED.

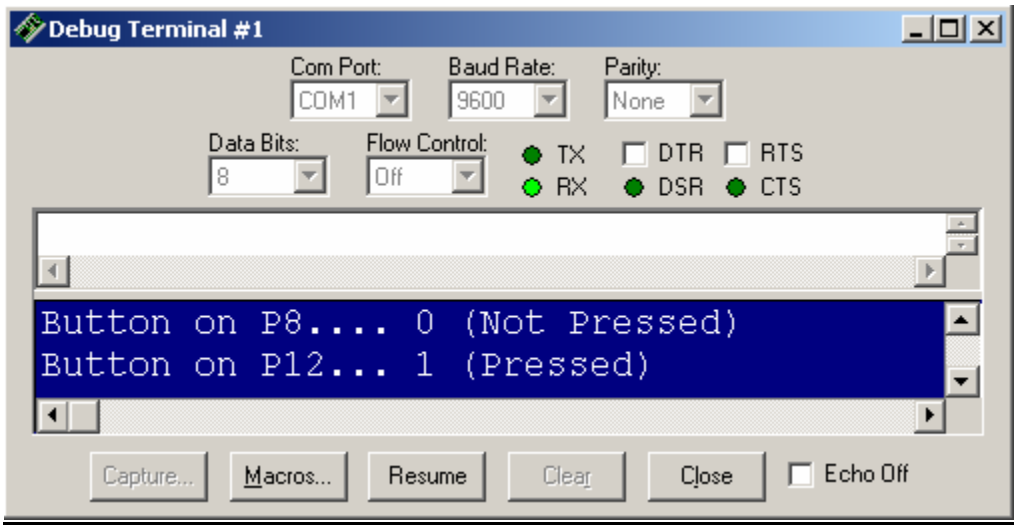



**Figure 67: DEBUG Output from Testing Pushbuttons**

## TESTING THE SEVEN-SEGMENT LED

Run the following code to check the seven-segment LED:

```
' =================================================================================
'
'   File....... HexCrawler Seven-Segment Display Test.BS2
'   Purpose.... Simple checkout of seven-segment display connections
'   Author..... Parallax
'   E-mail..... support@parallax.com
'   Started....
'   Updated.... 09 AUG 2003
'
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' =================================================================================


' -----[ Program Description ]---------------------------------------------
'
' Displays digits 0 through 9 on seven-segment display to test connections.
'
' Segment map:
'
'       (a)
'      -----
' (f) |     | (b)
'     | (g) |
'      -----
' (e) |     | (c)
'     |     |
'      -----
'       (d)


' -----[ Revision History ]------------------------------------------------


' -----[ I/O Definitions ]--------------------------------------------------

Segments        VAR     OUTL                    ' output on pins 0 - 7


' -----[ Constants ]-------------------------------------------------------

                        '.edcbafg               ' display segments
Dig0            CON     %01111110               ' segment data for digits
Dig1            CON     %00011000
Dig2            CON     %01101101
Dig3            CON     %00111101
Dig4            CON     %00011011
Dig5            CON     %00110111
Dig6            CON     %01110111
Dig7            CON     %00011100
Dig8            CON     %01111111
Dig9            CON     %00011111
```

```
DigA            CON     %01011111               ' hex A - F (10 - 15)
DigB            CON     %01110011
DigC            CON     %01100110
DigD            CON     %01111001
DigE            CON     %01100111
DigF            CON     %01000111


' -----[ Variables ]------------------------------------------------------

idx             VAR     Nib                     ' digit index


' -----[ EEPROM Data ]----------------------------------------------------


' -----[ Initialization ]--------------------------------------------------

Setup:
  DIRL = %01111111                              ' P0 - P6 are outputs


' -----[ Program Code ]----------------------------------------------------

Main:
  DO
    FOR idx = $0 TO $F                          ' display all digits
      LOOKUP idx, [Dig0, Dig1, Dig2, Dig3,
                   Dig4, Dig5, Dig6, Dig7,
                   Dig8, Dig9, DigA, DigB,
                   DigC, DigD, DigE, DigF], Segments
      PAUSE 500
    NEXT
  LOOP                                          ' loop forever


' -----[ Subroutines ]-----------------------------------------------------
```

If the circuit is wired correctly the BASIC Stamp 2 module will display digits 0 through 9 on the seven-segment LED.

## USER INTERFACE TEST

Run the following code to check the seven-segment LED with the pushbuttons:

```
' =============================================================================
'
'   File....... HexCrawler Interface Test.BS2
'   Purpose.... Test HexCrawler buttons and 7-Segment display
'   Author..... Parallax
'   E-mail..... support@parallax.com
'   Started....
'   Updated.... 11 AUG 2003
'
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' =============================================================================


' -----[ Program Description ]---------------------------------------------
'
' Scan HexCrawler buttons and update 7-segment display.  Button1 increments
' the display, Button2 decrements the display.  This program introduces a
' subroutine to scanning and debouncing both buttons without the use of the
' PBASIC BUTTON command.


' -----[ Revision History ]------------------------------------------------


' -----[ I/O Definitions ]-------------------------------------------------

ModeBtn         PIN     8                       ' select robot mode
StartBtn        PIN     12                      ' start/stop robot

Segments        VAR     OUTL                    ' output on pins 0 - 7


' -----[ Constants ]-------------------------------------------------------

                        '.edcbafg               ' display segments
Dig0            CON     %01111110               ' segment data for digits
Dig1            CON     %00011000
Dig2            CON     %01101101
Dig3            CON     %00111101
Dig4            CON     %00011011
Dig5            CON     %00110111
Dig6            CON     %01110111
Dig7            CON     %00011100
Dig8            CON     %01111111
Dig9            CON     %00011111

Pressed         CON     1                       ' button states
NotPressed      CON     0


' -----[ Variables ]-------------------------------------------------------
```

```
btns            VAR     Nib                     ' button holder
btn1            VAR     btns.BIT0               ' debounced button value
btn2            VAR     btns.BIT1               ' deboucned button value
idx             VAR     Nib                     ' digit index
counter         VAR     Nib                     ' current digit to display


' -----[ EEPROM Data ]----------------------------------------------------


' -----[ Initialization ]-------------------------------------------------

Setup:
  DIRL = %01111111                              ' P0 - P6 are outputs
  GOSUB Show Digit                              ' initialize display


' -----[ Program Code ]---------------------------------------------------

Main:
  DEBUG CLS,
        "Button & Display Test", CR, CR,
        "Mode (P8)........... ", CR,
        "Start/Stop (P12)... "

  DO
    GOSUB Get Buttons                           ' scan buttons
    GOSUB Show Buttons                          ' show states
    IF (btns > %00) AND (btns < %11) THEN       ' one or the other pressed?
      counter = counter + btn1 // 10            ' increment if Button1 = 1
      counter = counter + (9 * btn2) // 10      ' decrement if Button2 = 1
      GOSUB Show Digit                          ' update display
      PAUSE 250                                 ' 1/4 sec between changes
    ENDIF
  LOOP                                          ' do forever


' -----[ Subroutines ]----------------------------------------------------

' Show digit in "counter" on 7-segment display

Show_Digit:
  DEBUG CRSRXY, 0, 5, DEC ?counter              ' update DEBUG screen
  LOOKUP counter, [Dig0, Dig1, Dig2, Dig3, Dig4,
                   Dig5, Dig6, Dig7, Dig8, Dig9], Segments
  RETURN


' Scan and debounce both buttons

Get Buttons:
  btns = %0011                                  ' assume both pressed
  FOR idx = 1 TO 5
    btns.BIT0 = btns.BIT0 & ModeBtn             ' scan mode button
    btns.BIT1 = btns.BIT1 & StartBtn            ' scan start/stop button
    PAUSE 5                                     ' debounce delay
  NEXT
  RETURN
```

```
' Show current button states

Show Buttons:
  FOR idx = 0 TO 1
    DEBUG CRSRXY, 20, 2 + idx                  ' move cursor
    IF (btns.LOWBIT(idx) = Pressed) THEN       ' check and display status
      DEBUG "Pressed", CLREOL
    ELSE
      DEBUG "Not Pressed"
    ENDIF
  NEXT
  RETURN
```

## TUNING THE HEXCRAWLER'S LEGS

The HexCrawler_Home.bs2 program is a routine designed to assist in centering up the servos.

**CAUTION**: Please ensure that your HexCrawler is mounted on top of a box or elevated platform with its legs clear of your working surface before you run this program.

```
'---- [Adjust HexCrawler Legs] ------------------------------------
'{$STAMP BS2}
'{$PBASIC 2.5}
' Ver 1.0
'
'   File....... HexCrawler Home.BS2
'   Purpose.... Place HexCrawler legs centered and down
'   Author..... Parallax
'   E-mail..... support@parallax.com
'   Started.... 16 April 2004
'   Updated.... 18 April 2004
'
'
'
'
'
'-------------------------------------------------------------------

' -----[ I/O Definitions ]--------------------------------------
PSC             PIN     15              ' PSC module

' -----[ Constants ]-------------------------------------------
LiftRamp        CON     1               ' Very fast leg lift
HorzCenter      CON     750             ' horizontal servo leg center

' -----[ Variables ]-------------------------------------------
servoAddr       VAR     Byte            ' Servo addresses
horzPos0        VAR     Word            ' Horizontal leg positions
horzPos1        VAR     Word
Baud            VAR     Word            ' Buad setting for PSC
Buff            VAR     Byte(3)         ' PSC response


'---- [Initialize Leg positions] ---------------------------------
horzPos0 = HorzCenter
horzPos1 = HorzCenter


Fix_Baud:
Chk2k4:
  Baud = 396 + $8000
  DEBUG "Checking PSC with  2400 Baud",CR
  SEROUT PSC, Baud, ["!SCVER?",CR]
  SERIN  PSC, Baud, 500,Chk38k4 , [STR buff\3]
  DEBUG  "Found PSC version: ", buff(0), buff(1), buff(2), CR
JackUptheBaud:
  DEBUG "Setting Baudrate to 38k4",CR
```

```
  SEROUT PSC,Baud,["!SCSBR",1,CR]
  SERIN  PSC,6,300,Chk2k4,[STR buff\3]
  DEBUG "Baud reply:  ", buff(0), buff(1),DEC1 buff(2), CR
  Baud = 6 + $8000
SBcont:
  GOTO Fix_Baud

Chk38k4:
  Baud = 6 + $8000
  DEBUG "Checking PSC with 38400 Baud",CR
  SEROUT PSC, Baud, ["!SCVER?",CR]
  SERIN  PSC, Baud, 500,Chk38k4 , [STR buff\3]
  DEBUG  "Found PSC version: ", buff(0), buff(1), buff(2), CR

'Place legs in center down position
Home Legs:
FOR servoAddr = 1 TO 5 STEP 2
  SEROUT PSC, Baud+$8000,["!SC", servoAddr, LiftRamp, $2C, $01, CR]
  PAUSE 500
  SEROUT PSC, Baud+$8000,["!SC", servoAddr-1,LiftRamp,horzPos0.LOWBYTE,horzPos0.HIGHBYTE, CR]
  PAUSE 500
  SEROUT PSC, Baud+$8000,["!SC", servoAddr, LiftRamp, $B0, $04, CR]
NEXT
FOR servoAddr = 7 TO 11 STEP 2
  SEROUT PSC, Baud+$8000,["!SC", servoAddr, LiftRamp, $B0, $04, CR]
  PAUSE 500
  SEROUT PSC, Baud+$8000,["!SC", servoAddr-1,LiftRamp,horzPos1.LOWBYTE,horzPos1.HIGHBYTE, CR]
  PAUSE 500
  SEROUT PSC, Baud+$8000,["!SC", servoAddr,LiftRamp, $2C, $01, CR]
NEXT
DEBUG CR, CR, "Ready to Adjust Legs"
END
```

### Adjusting the HexCrawler's Legs

Not all servos are created equal. Therefore, you might have to fine-tune the legs. After the program executes, it leaves the robot in the home position: centered horizontally and standing. When looking down at the legs they should appear perpendicular to the body. Don't worry if you can't get all the legs perfectly perpendicular. Being off a degree or two will not adversely affect the robot's gait. Refer to the horizontal and vertical adjustment information on the next two pages to adjust the legs.

### Horizontal Servo Adjustment



1. Loosen (do not remove) the side support screws on the same side as the leg as well as the lower front and rear support screws. (see Figure 68)

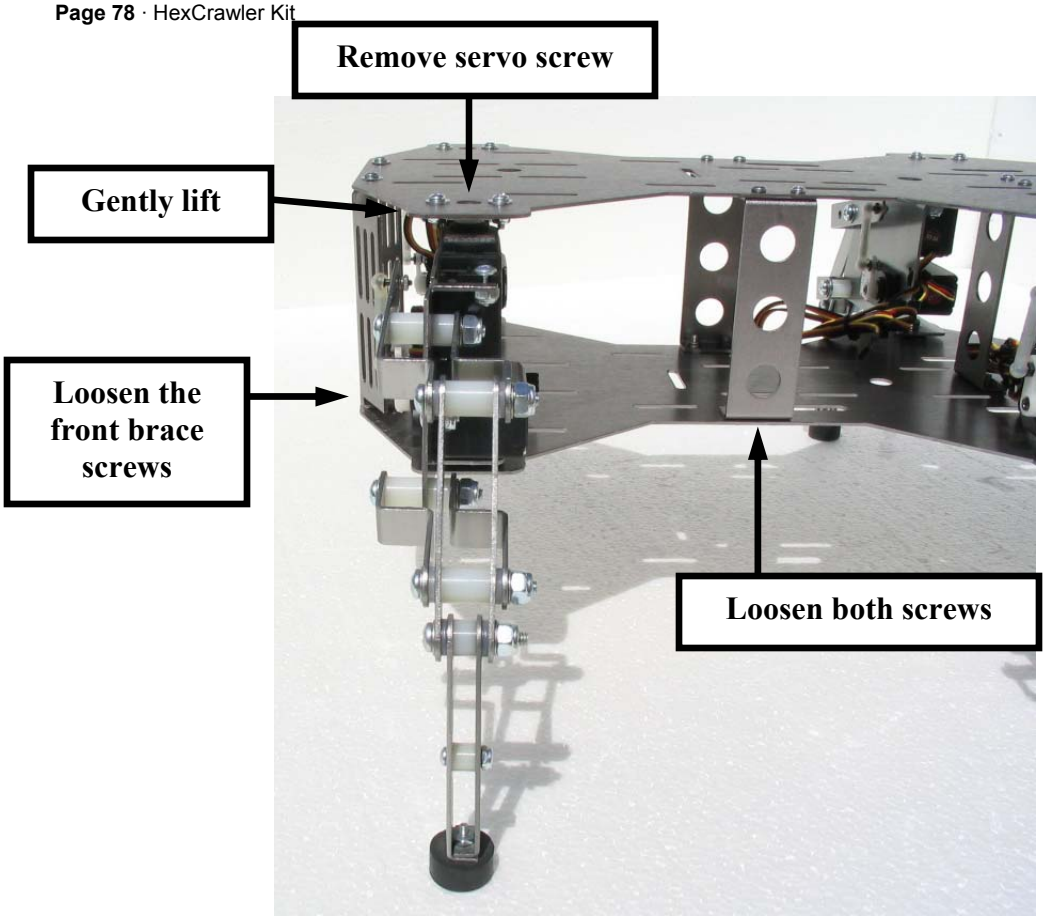
2. Remove the top horizontal servo screw.

3. Gently lift the lip of the top half of the servo body to pop the servo spindle out of its holder and rotate it so that the leg is as close to straight out from the body as possible.

4. Re-install the top horizontal servo screw.

5. Perform steps 1 – 4 for each leg that is not centered.

6. Re-tighten the side, front and rear support screws.




**68: Horizontal Leg Adjustment**

## Vertical Servo Adjustment

1. Loosen the servo screw on the servo control horn and remove the servo control horn. (Figure 69)
2. Turn the servo control horn until it is aligned as close to 90 degrees vertical as possible.
3. Re-install the servo screw
4. Repeat steps 1- 3 for every leg that needs adjustment.

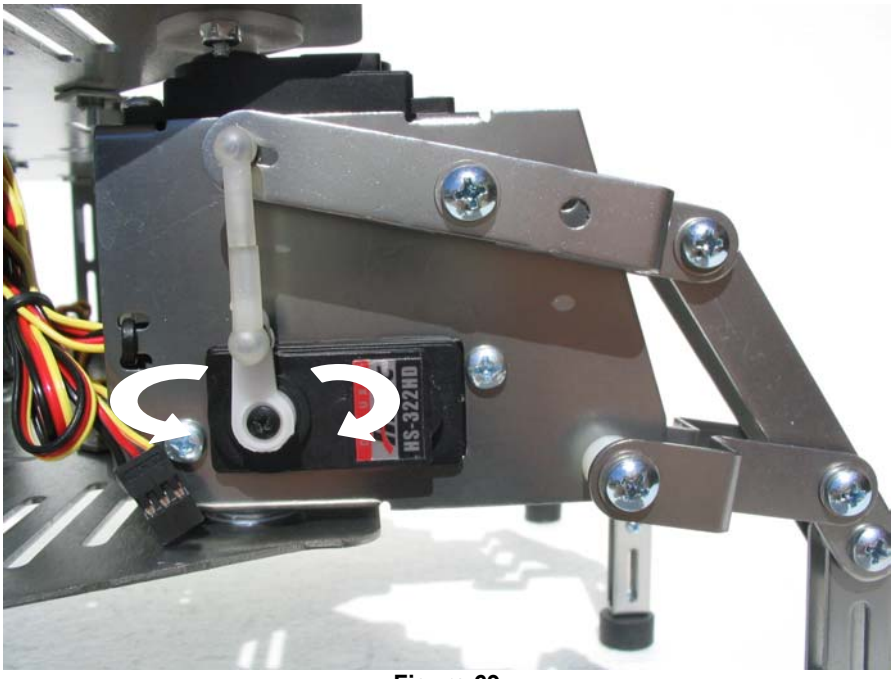



**Figure 69**

The next procedure will ensure that all of the legs are evenly sharing the weight of the HexCrawler

Once all of the servo arms have been adjusted as vertical as possible, place your HexCrawler on a smooth surface in a standing position. One leg at a time swing the legs forward and backwards and note any differences in resistance between the leg and the floor.

For legs that appear to be bearing more of the weight than the other legs on the HexCrawler - Remove the dog bones from the leg and twist the dog bones closer together. Re-install the dog bone and re-test.

For leg that appear to be bearing less of the weight than the other leg of the HexCrawler - Remove the dog bones from the leg and twist the dog bones further apart - Re-install the dog bone and re-test.

Once all of the legs have been adjusted, you are ready to load and run the walking program.

## HEXCRAWLER WALKING PROGRAM

```
'---- [HexCrawler: Forward, Backward, Left, and Right with Ramp] -----
'{$STAMP BS2}
'{$PBASIC 2.5}
' Ver 1.0
'
'   File....... HexCrawler PSC Button.BS2
'   Purpose.... Select 15 different gaits
'   Author..... Parallax
'   E-mail..... support@parallax.com
'   Started.... 16 April 2004
'   Updated.... 18 April 2004
'
' HexWalker PSC Button.bs2 contains 15 different gait settings.
'
' Button operation:
' Press the up and down buttons to select gaits.
'
' Press both buttons to accept selection
' Press both buttons during program execution to enter selection mode.
'
' If Home is selected (0) the robot will center and lower it's legs
' for adjustment.  Press the reset button on the BOE to restart.
'
' Code Sections:
' The Button Code Section can be replaced with your own code for use
' with devices like RC units or various sensors.
'
' The Compare Gaits routine uses the "selectedGait" variable to determine
' direction and turning.
'
' Use the "selectedGait" variable to pass gait information
' to the "Compare_Gait" routine.
'
' Display
' (0)   $00 - Home
' (1)   $01 - Spin Left
' (2)   $02 - Spin Right
'
' Display                          Display
' (3)   $10 - Forward Fast         (6)   $20 - Forward
' (4)   $11 - Fast Forward Left    (7)   $21 - Forward Left
' (5)   $12 - Fast Forward Right   (8)   $22 - Forward Right
'
' Display                          Display
' (9)   $20 - Backward             (C)   $40 - Fast Backward
' (A)   $21 - Backward Left        (D)   $41 - Fast Backward Left
' (B)   $22 - Backward Right       (E)   $42 - Fast Backward Right
'                                  (F)   Open
'
'
' 7 Segement LED Display:
'    Segment map:      .edc bafg           .edc bafg
'       (a)         0 %0111 1110  $7E   8 %0111 1111  $7F
'      -----        1 %0001 1000  $18   9 %0001 1111  $1F
' (f) |     | (b)   2 %0110 1101  $6D   A %0101 1111  $5F
'     | (g) |       3 %0011 1101  $3D   B %0111 0011  $73
```

```
'      -----         4 %0001 1011  $1B   C %0110 0110  $66
' (e) |     | (c)    5 %0011 0111  $37   D %0111 1001  $79
'     |     |        6 %0111 0111  $77   E %0110 0111  $67
'      -----         7 %0001 1100  $1C   F %0100 0111  $47 - Open
'       (d)
'
' Adjustable Values:
' You will find the values below can be changed to increase
' or decrease the speed and turning radius of you HexCrawler.
'
' Speed           CON     15      ' Step value in For...Next Loops
' hROM            CON     250     ' Horizontal range of motion
' LiftRamp        CON     1       ' Lift/Lower legs fast
' Fast            CON     $A      ' Adjustable from
' Medium          CON     $D        $1 - Very Fast to
' Slow            CON     $F        $F(15) - Very Slow
' Pause Lower     CON     150     ' Give legs time to lower
' Pause_Stride    CON     1       ' Increase to slow legs
'                                   Increasing to values above 10
'                                   can cause jittery legs depending
'                                   on ramp and hROM
'
' The only constant that might cause issues is hROM.
' hROM is the leg's horizontal range of motion.
' Increasing this value above 300 can cause the
' legs to slap together. hROM is dependent on ramp
' and speed values.
'
' Ramp is the speed at which a servo moves
' to a new position. Ramp is adjustable from
' $1 - Very Fast to
' $F (15) - Very Slow
' If ramp is very fast the legs might not have enough
' time to raise and lower.
' Ramp is dependent on speed and hROM.
'
' Speed is the step value used in
' For...Next loops.  Decreasing this value below
' 8 can cause very slow leg movements.  Conversely,
' large values can cause jittery legs.
' And you guessed it... Speed is dependent
' on hROM and Ramp
'
' Lastly, pauses allow the servos time to finish
' their movements.  You might have to adjust
' pause values if you change hROM, Ramp, or Speed.
'
'
' Happy Programming !!!
'
'-------------------------------------------------------------------------


' -----[ I/O Definitions ]--------------------------------------------------
PSC             PIN     15              ' PSC module
ModeBtn         PIN     8               ' select robot mode up
StartBtn        PIN     12              ' select robot mode down
Segments        VAR     OUTL            ' output on pins 0 - 7
```

```
'---- [Walking Variables] ------------------------------------------------
counter         VAR     Word            ' counter in FOR..NEXT loops
horzPos0        VAR     Word            ' Horizontal leg positions
horzPos1        VAR     Word
direction       VAR     Word            ' reverse horizontal leg movements
Baud            VAR     Word            ' Buad setting for PSC
Buff            VAR     Byte(3)         ' holds PSC response
selectedMode    VAR     Byte            ' Mode selected from RC stick position
currentMode     VAR     Byte            ' Current walking mode
servoAddr       VAR     Byte            ' Servo addresses
RightRamp       VAR     Nib             ' Right/Left leg speed
LeftRamp        VAR     Nib
selectedLegs    VAR     Nib             ' legs to raise or lower
flag            VAR     Bit             ' flag

'---- [Button Variables]-------------------------------------------------
btns            VAR     Nib             ' button holder
btn1            VAR     btns.BIT0       ' debounced button value
btn2            VAR     btns.BIT1       ' deboucned button value
idx             VAR     Nib             ' digit index
btnIndx         VAR     Nib             ' current digit to display
'---- [button states] ----------------------------------------------------
Pressed         CON     1
NotPressed      CON     0

'----- [Adjustable values] ------------------------------------------------
' Increasing hROM may cause the legs to hit
' each other.
hROM            CON     250             ' Horizontal Range of Motion
LiftRamp        CON     1               ' Very fast leg lift/lower

'---- [Adjutstable Ramp values] -----------------------------------------
' Do a little experimenting
Speed           CON     15      ' Step value in For Loops
Fast            CON     $A      ' Adjustable from $1 - Very Fast
Medium          CON     $D      ' to $F(15) - Very Slow
Slow            CON     $F
Pause Lower     CON     150     ' Give legs time to lower
Pause Stride    CON     1       ' Increase to slow legs
                                ' Increasing to values above 10
                                ' can cause jittery legs depending
                                ' on ramp and hROM
'---- [Walking Constants] ------------------------------------------------
HorzCenter      CON     750     ' horizontal servo leg center
Forward         CON     1       ' Walk forward
Backward        CON     -1      ' Walk backward
TurnLeft        CON     -1
TurnRight       CON     1
Legs1_3_5       CON     0       ' Access Legs 1, 3, and 5
Legs2_4_6       CON     2


'---- [EEPROM Data] -----
' 7 Segment LED and Gaits
' Hex      0   1   2   3   4   5   6   7   8   9   A   B   C   D   E    F
     DATA $7E,$18,$6D,$3D,$1B,$37,$77,$1C,$7F,$1F,$5F,$73,$66,$79,$67',$47
Gait DATA $00,$01,$02,$10,$11,$12,$20,$21,$22,$30,$31,$32,$40,$41,$42
```

```
'----[Setup PSC for 38K4 BAUD] ----------
'DEBUG "Here we go !!!", CR
Fix Baud:
Chk2k4:
  Baud = 396 + $8000
  'DEBUG "Checking PSC with  2400 Baud",CR
  SEROUT PSC, Baud, ["!SCVER?",CR]
  SERIN  PSC, Baud, 500,Chk38k4 , [STR buff\3]
  'DEBUG  "Found PSC version: ", buff(0), buff(1), buff(2), CR
JackUptheBaud:
  'DEBUG "Setting Baudrate to 38k4",CR
  SEROUT PSC,Baud,["!SCSBR",1,CR]
  SERIN  PSC,6,300,Chk2k4,[STR buff\3]
  'DEBUG "Baud reply:  ", buff(0), buff(1),DEC1 buff(2), CR
  Baud = 6 + $8000
SBcont:
  GOTO Fix Baud

Chk38k4:
  Baud = 6 + $8000
  'DEBUG "Checking PSC with 38400 Baud",CR
  SEROUT PSC, Baud, ["!SCVER?",CR]
  SERIN  PSC, Baud, 500,Chk38k4 , [STR buff\3]
  'DEBUG  "Found PSC version: ", buff(0), buff(1), buff(2), CR

'DEBUG "Baud Set", CR

DIRL = %01111111              ' P0 - P6 are outputs
currentMode = $00             ' Assume Home Mode
'speed = 15
GOTO Update_Button_Index      ' Select a gait at reset

'---- [Debug Screen Setup] -----
Setup:
  DEBUG CLS,
        "Button, Display, and Mode Settings", CR, CR,
        "Up Button (P8).... ", CR,
        "Down Button (P12). ", CR,CR,CR,
        "Current Mode......",CR,
        "Selected Mode.....",CR,
        "LED Display.......",CR
  GOSUB Show_Buttons

  READ btnIndx, Segments                      ' display current gait
  'DEBUG HEX ?btnIndx                                 ' For loop step value
  direction = Forward                         ' Initialize Leg positions
  horzPos0 = HorzCenter + (Hrom/2)            ' and direction
  horzPos1 = HorzCenter - (Hrom/2)
  RightRamp = Fast                            ' Assign leg ramp
  LeftRamp = Fast
  GOSUB Raise                                 ' Raise legs 1, 3, and 5
  GOSUB Stride Turn                           ' Move legs Horizontally
  GOSUB Lower                                 ' Lower legs 1, 3, and 5

'---- [Button Code Section] ------
Update Button Index:
'DEBUG "Select Gait",CR
PAUSE 10
    IF flag = 0 THEN                          ' Positin legs if reset
```

```
      flag = 1                                ' Wait for gait selection
      GOTO Setup
    ELSE
      flag = 0
    ENDIF
  DO
  DEBUG CRSRXY, 0, 10,"Gait selection = Up/Down Buttons",CR,
                      "Enter/New Gait = Both"
    GOSUB Get Buttons                         ' scan buttons
    GOSUB Show Buttons
    IF (btns > %00) AND (btns < %11) THEN     ' one or the other pressed?
      btnIndx = btnIndx + btn1 // 15          ' increment if Button1 = 1
      btnIndx = btnIndx + (14 * btn2) // 15   ' decrement if Button2 = 1
      ' Read EEPROM
      READ btnIndx, Segments                  ' Update LED display
      READ gait+btnIndx,selectedMode          ' Update Selected gait Mode
      PAUSE 250                               ' 1/4 sec between changes
      GOSUB Show_Buttons
    ENDIF
    IF (btns.BIT0=Pressed) AND (btns.BIT1=Pressed) THEN
      ' Exit LOOP
      GOSUB Show Buttons
      GOTO Compare Gaits
    ENDIF
  LOOP
'---- [Display Button and gait information to Debug Screen] -----
 Show Buttons:
  FOR idx = 0 TO 1
    DEBUG CRSRXY, 18, 2 + idx                 ' move cursor
    IF (btns.LOWBIT(idx) = Pressed) THEN      ' check and display status
      DEBUG "Pressed", CLREOL
    ELSE
      DEBUG "Not Pressed"
    ENDIF
  NEXT
    DEBUG CRSRXY, 18, 6, HEX currentMode.HIGHNIB, HEX currentMode.LOWNIB
    DEBUG CRSRXY, 18, 7, HEX selectedMode.HIGHNIB, HEX selectedMode.LOWNIB
    DEBUG CRSRXY, 18, 8, HEX btnIndx
  RETURN                                      '
'---- [End of Button Code Section] ------

'---- [Compare current gait to selected gait] -----
Compare_Gaits:
'DEBUG HEX ?selectedMode
SELECT selectedMode
  CASE $00                                    ' Home legs
    direction = Forward                       ' Initialize Leg positions
    horzPos0 = HorzCenter
    horzPos1 = HorzCenter
    RightRamp = Fast                             ' Assign leg ramp
    LeftRamp = Fast
    GOTO Home Legs
'************************************
' Continue if the current gait is equal to the
' selected gait.  This section might be
' necessary if you are using a sensor
' or RC unit to select gaits.
  'CASE currentMode
    'IF selectedMode > $02 THEN
```

```
      'DEBUG "Continue Tripod",CR
      'GOTO Continue Tripod Gait
    'ELSE
      'DEBUG "Continue TriTurn Mode",CR
      'GOTO Continue Tripod Turn
    'ENDIF
'************************************
ENDSELECT

' Spin selected
IF selectedMode <= $02 THEN
'DEBUG "Tripod Turn",CR
' Setup Ramp
  RightRamp = Fast
  LeftRamp = RightRamp
  currentMode = selectedMode
  'Spin Left
  IF selectedMode.LOWNIB = $1 THEN
    horzPos0 = HorzCenter - (Hrom/2)
    horzPos1 = HorzCenter + (Hrom/2)
    direction = TurnLeft
    'DEBUG "Tripod Left",CR
    GOTO Tripod Turn
  ELSE
    'Spin Right
    horzPos0 = HorzCenter + (Hrom/2)
    horzPos1 = HorzCenter - (Hrom/2)
    direction = TurnRight
    'DEBUG "Tripod Right",CR
    GOTO Tripod_Turn
  ENDIF
ENDIF
' Check selected mode HighNib
' and compare to current direction.
  SELECT selectedMode.HIGHNIB          ' Fast Forward
    CASE $1
      'DEBUG "Fast Forward ",CR
      RightRamp = Fast                 ' Set Fast Walk
      ' Setup leg ramp for
      ' gradual OR straight turn
      GOSUB Set Ramp
      GOSUB Check_Forward_Direction    ' Check Crawler current direction
    CASE $2                            ' Forward
      'DEBUG "Forward",CR
      RightRamp = Medium               ' Set Medium Walk
      GOSUB Set Ramp
      GOSUB Check Forward Direction
    CASE $3                            ' Backward
      'DEBUG "Backward ",CR
      RightRamp = Medium
        GOSUB Set Ramp
        GOSUB Check Back Direction
    CASE $4                            ' Fast Backward
      RightRamp = Fast
      'DEBUG "Fast Backward ",CR
        GOSUB Set Ramp
        GOSUB Check Back Direction
  ENDSELECT
```

```
Set_Ramp:
' Check LowNib to determine leg ramp
' 0 = straight
' 1 = slow left side
' 2 = slow right side
  SELECT selectedMode.LOWNIB
    CASE $0
      'DEBUG "Straight ", CR
      LeftRamp = RightRamp
    CASE $1
      'DEBUG "left ", CR
      LeftRamp = Slow
    CASE $2
      'DEBUG "Right ", CR
      LeftRamp = RightRamp
      RightRamp = Slow
  ENDSELECT
  'DEBUG HEX ?RightRamp, HEX ?LeftRamp
RETURN

' Forward selected
' If the bot was walking backward change
' direction to forward. If the bot
' was walking forward continue
Check_Forward_Direction:
  IF direction = Backward THEN
    direction = -direction
    currentMode = selectedMode
    'DEBUG "Changed to Forward", CR
    GOTO TriPod_Gait
  ELSE
    currentMode = selectedMode
    'DEBUG "Continue Forward",CR
    GOTO Continue tripod gait
  ENDIF
RETURN

' Backward selected
' If the bot was walking forward change
' direction to backward. If the bot
' was walking backward continue
Check_Back_Direction:
  IF direction = Forward THEN
    direction = -direction
    currentMode = selectedMode
    'DEBUG "Changed to Forward", CR
    GOTO TriPod Gait
  ELSE
    currentMode = selectedMode
    'DEBUG "Continue Forward",CR
    GOTO Continue tripod gait
  ENDIF
RETURN

'DEBUG HEX ?selectedMode, CR, HEX ?selectedMode.HIGHNIB, CR
'---- [End of Compare gaits] -----

' ----- [Tripod Main Walk Routine] ------------------------------------------------
Tripod_Gait:
```

```
  DEBUG CLS, "Walk"
  selectedLegs = Legs1 3 5
  GOSUB Raise              ' Raise legs 1, 3, and 5
  GOSUB Stride Turn        ' Move legs Horizontally
  GOSUB Lower              ' Lower legs 1, 3, and 5
' Check if both buttons are pressed
' This is where you would place your
' own code for use with other devices.
' At this point the legs can change
' direction and ramp values without reseting.
  GOSUB Get Buttons
  IF (btns.BIT0=Pressed) AND (btns.BIT1=Pressed) THEN
    DEBUG CLS, "Reset",CR
    GOTO Update Button Index
  ENDIF
Continue Tripod Gait:
'DEBUG "TriPod Left Ramp = ", ?LeftRamp, "Right Ramp = ", ?RightRamp
  selectedLegs = Legs2_4_6
  GOSUB Raise              ' Raise legs 2, 4, and 6
  direction = -direction
  GOSUB Stride Turn
  GOSUB Lower
  direction = -direction
GOTO Tripod_Gait

' ----- [Tripod Main Turn] ------------------------------------------------
' This routine is similar to TriWalk
Tripod Turn:
  DEBUG CLS, "Spin"
  selectedLegs = Legs2_4_6
  GOSUB Raise
  GOSUB Rotate       ' Turn
  GOSUB Lower
' Check if both buttons are pressed
' This is where you would place your
' own code for use with other devices.
' At this point the legs can change
' direction and ramp values without reseting.
  GOSUB Get Buttons
  IF (btns.BIT0=Pressed) AND (btns.BIT1=Pressed) THEN
    DEBUG CLS, "Reset",CR
    GOTO Update_Button_Index
  ENDIF
Continue Tripod Turn:
'DEBUG "TriPod Turn Left Ramp = ", ?LeftRamp, "Right Ramp = ", ?RightRamp
  selectedLegs = Legs1 3 5
  GOSUB Raise
  direction = -direction
  GOSUB Rotate
  direction = -direction
  GOSUB Lower
GOTO Tripod Turn

' ----- [TriPod Raise Legs] -----------------------------------------------
Raise:
FOR servoAddr = (1+selectedLegs) TO (9+selectedLegs) STEP 4
  IF (servoAddr<6) THEN
    SEROUT PSC,Baud+$8000,["!SC",servoAddr, LiftRamp, 26, 01, CR]
  ELSE
```

```
    SEROUT PSC,Baud+$8000,["!SC",servoAddr, LiftRamp, B0, 04, CR]
  ENDIF
NEXT
' Adjustable
'PAUSE 100
RETURN

' ----- [TriPod Lower Legs] ------------------------------------------------
Lower:
FOR servoAddr = (1+selectedLegs) TO (9+selectedLegs) STEP 4
  IF (servoAddr<6) THEN
    SEROUT PSC,Baud+$8000,["!SC",servoAddr, LiftRamp, B0, 04, CR]
  ELSE
    SEROUT PSC,Baud+$8000,["!SC",servoAddr, LiftRamp, 26, 01, CR]
  ENDIF
NEXT
' Adjustable
PAUSE Pause_lower
RETURN


' ----- [Forward/Back horizontal leg movement] -----------------------------
' Update horizontal leg array values
' step through horizontal servo address
' If servo address is a middle leg
' move in the opposite direction
Stride Turn:
FOR counter = 0 TO hROM STEP speed
'DEBUG ?counter
horzPos0 = horzPos0 - (speed*direction)
horzPos1 = horzPos1 + (speed*direction)
FOR servoAddr = 0 TO 4 STEP 2
    IF (servoAddr=2)THEN
      SEROUT PSC,Baud+$8000,["!SC",servoAddr,RightRamp,horzPos0.LOWBYTE,horzPos0.HIGHBYTE,
CR]
    ELSE
      SEROUT PSC,Baud+$8000,["!SC",servoAddr,RightRamp,horzPos1.LOWBYTE,horzPos1.HIGHBYTE,
CR]
    ENDIF
 NEXT
 FOR servoAddr = 6 TO 10 STEP 2
    IF (servoAddr=8)THEN
      SEROUT PSC,Baud+$8000,["!SC",servoAddr,LeftRamp,horzPos0.LOWBYTE,horzPos0.HIGHBYTE, CR]
    ELSE
      SEROUT PSC,Baud+$8000,["!SC",servoAddr,LeftRamp,horzPos1.LOWBYTE,horzPos1.HIGHBYTE, CR]
    ENDIF
 NEXT
 'Adjustable
 PAUSE Pause_Stride
NEXT
RETURN

' ----- [Left/Right horizontal leg movement] -------------------------------
' Similar to the Stride sub routine
' but legs 2, 4, and 6 and 1, 3, 5 move in the
' same direction
Rotate:
FOR counter = 1 TO hROM STEP speed
 horzPos0 = horzPos0 - (speed*direction)
```

```
 horzPos1 = horzPos1 + (speed*direction)
  FOR servoAddr = 0 TO 10 STEP 2
    IF (servoAddr=2) OR (servoAddr=6) OR (servoAddr=10) THEN
      SEROUT PSC,Baud+$8000,["!SC",servoAddr,RightRamp,horzPos0.LOWBYTE,horzPos0.HIGHBYTE,
CR]
    ELSE
      SEROUT PSC,Baud+$8000,["!SC",servoAddr,RightRamp,horzPos1.LOWBYTE,horzPos1.HIGHBYTE,
CR]
    ENDIF
  NEXT
NEXT
 PAUSE Pause_Stride
RETURN

' ----- [Read Buttons] -----------------------------
Get Buttons:
  btns = %0011                                  ' assume both pressed
  FOR idx = 1 TO 5
    btns.BIT0 = btns.BIT0 & ModeBtn             ' scan mode button
    btns.BIT1 = btns.BIT1 & StartBtn            ' scan start/stop button
    PAUSE 5                                     ' debounce delay
  NEXT
RETURN

' ----- [Home] -------------------------------------
'Place legs in starting position
'Initialize Tripod leg Positions:
'DEBUG "Initialize Tripod leg Positions",CR
Home Legs:
FOR servoAddr = 1 TO 5 STEP 2
  SEROUT PSC, Baud+$8000,["!SC", servoAddr, LiftRamp, $2C, $01, CR]
  PAUSE 500
  SEROUT PSC, Baud+$8000,["!SC", servoAddr-1,LiftRamp,horzPos0.LOWBYTE,horzPos0.HIGHBYTE, CR]
  PAUSE 500
  SEROUT PSC, Baud+$8000,["!SC", servoAddr, LiftRamp, $B0, $04, CR]
NEXT
FOR servoAddr = 7 TO 11 STEP 2
  SEROUT PSC, Baud+$8000,["!SC", servoAddr, LiftRamp, $B0, $04, CR]
  PAUSE 500
  SEROUT PSC, Baud+$8000,["!SC", servoAddr-1,LiftRamp,horzPos1.LOWBYTE,horzPos1.HIGHBYTE, CR]
  PAUSE 500
  SEROUT PSC, Baud+$8000,["!SC", servoAddr,LiftRamp, $2C, $01, CR]
NEXT
DEBUG CLS, "Ready to Adjust Legs"
END
```