Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

Motor Mount and Wheel Kit (#27971) with Position Controller (#29319)

Features

- Powerful DC motors for plenty of torque
  ~150 RPM @ 12.0 VDC, 1.50 A, no load
  ~190 RPM @ 14.5 VDC, 1.60 A, no load
- Precision machined 6061 aluminum hardware
- Conveniently positioned screw holes make mounting this kit a breeze.
- Rugged pneumatic tires are well-suited for a variety of terrains
- +5 volt Supply for Position Controllers
- 36 encoder positions per revolution; approx 0.5 inch resolution with 6" wheel
- Compatible with any microcontroller
- Single I/O line can control up to 4 Position Controllers
- Strong but light: only 3.2 lbs (1.45 kg) per wheel assembly (kit contains two).

Application Ideas

- Sturdy and stylish midsize robot platforms
- Accurate distance tracking
- Autonomous exploration and data collection
- Part of a robust navigation system when incorporated with GPS and compass modules

General Description

The Motor Mount and Wheel Kit combines powerful 12 VDC motors and precision machined aluminum hardware to provide the power, strength, and beauty demanded by midsized robots. All custom parts are CNC machined at Parallax headquarters in Rocklin, California from 6061 billet aluminum. Conveniently positioned screw holes on the top, bottom, and front face of the bearing blocks provide a variety of quick and easy mounting options. The included 6 inch (15.3 cm) pneumatic rubber tires are durable enough to handle a variety of smooth or rugged terrains without hesitation.

The Position Controllers use a quadrature encoder system to reliably track the position and speed of each wheel at all times. With the included plastic encoder disks, each Position Controller has a resolution of 36 positions per rotation; this equates to approximately 0.5 inches of linear travel per position using the included 6 inch tires. The Position Controllers calculate and report position and average speed data on command. This leaves the main processor free to handle more important tasks like reading GPS coordinates, processing sensor information, and maneuvering complex environments.

The Position Controllers are compatible with any microcontroller via a single-wire half-duplex asynchronous serial communication (UART) bus. Up to four Position Controller devices can be controlled on the same bus to minimize I/O requirements.

For increased functionality, Position Controllers can be interfaced with HB-25 motor controllers (#29144; sold separately) to control the position of the wheel and automatically provide smooth speed ramping as well as accurate position advancement capability.

Table of Contents

Bill of Materials

Part #	Description		Quantity
27958	Motor, 12 VDC Left Hand		1
27959	Motor, 12 VDC Right Hand		1
34010	Motor Bearing Block – 6061 Aluminum		2
34015	Wheel, main rim section – 6061 Aluminum		2
34016	Wheel, rim ring section – 6061 Aluminum		2
34021	Axle – 6061 Aluminum		2
700-00100	1/8” x 1” Roll Pin		2
710-00015	4-40 x ½” Flat Head Screw		12
710-00105	¼”-20 x 5/8” Button Cap Screw		2
710-00110	M4 x 50 mm Socket Cap Screw		6
713-00012	3/8”-OD, #8-ID x 1” Spacer		6
713-00013	Cap, plastic valve stem		2
721-00004	Inner Tube, 6 x 1-1/4” Pneumatic		2
721-00006	Knobby Tire, 6 x 1-1/4” Rubber		2
724-00001	0.5” ID x 1.125” OD Ball Bearing		2
29319	Position Controller; each includes:		2
-	Position Controller PCB	1	-
-	Plastic Encoder Disk	1	-
-	4-40 x 3/16” Flat Head Screw	2	-
-	3-pin Servo Ext. Cable – 14”	1	-

Assembly Instructions

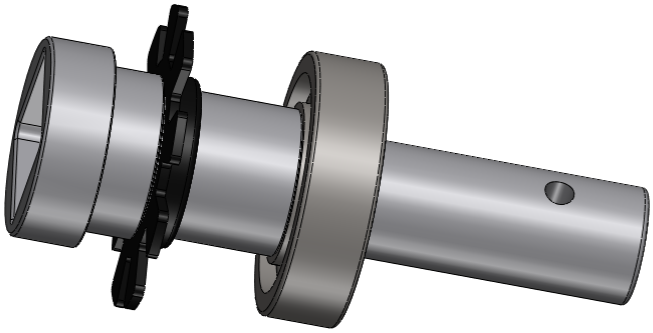
The following directions apply to both right-hand and left-hand assemblies.

Tools Required (not included)

- Size 1 Philips-tip screwdriver
- 3 mm Hex key
- 5/32″ Hex key
- Hammer
- Wood block
- Dish soap
- Tire pump

Step 1: Slide encoder disk and bearing on axle

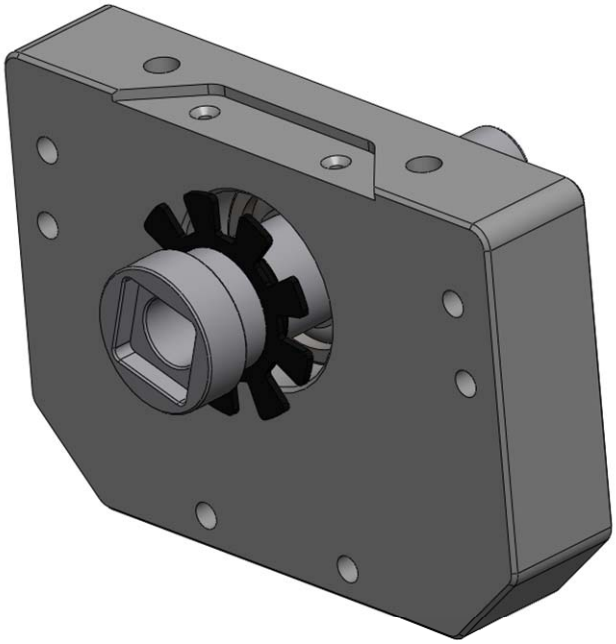
Slide the plastic encoder disk for the Position Controller on the axle with the tapered sleeve oriented away from the larger axle end. Then slide the bearing onto the axle.



Step 2: Insert bearing into mounting block

Carefully slide the axle and bearing assembly into the bearing pocket of the mounting block.

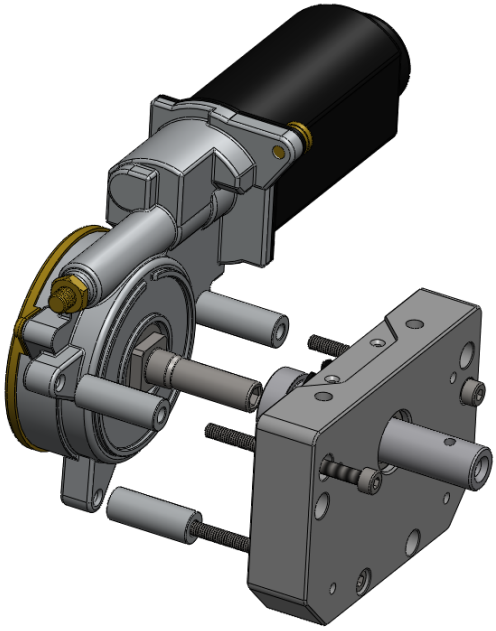
Important: Do not force the bearing into the bearing pocket. The pocket is precision machined to fit the bearing exactly. If the bearing does not slide in with delicate pressure, slightly adjust the insertion angle and try again. Using excessive force can cause the bearing to bind in the pocket and become very difficult to insert.



Step 3: Align axle and assemble mounting block with motor

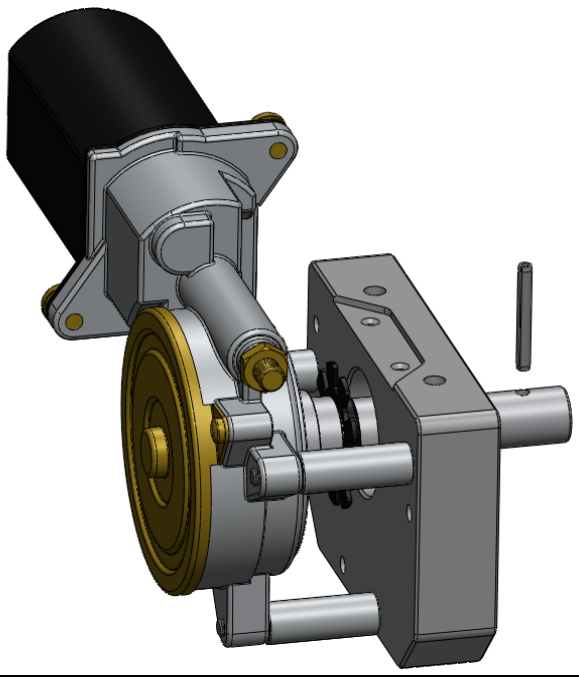
Align the drive shaft of the motor with the corresponding square cutout in the axle. If necessary, turn the axle by hand until it meshes with the motor shaft.

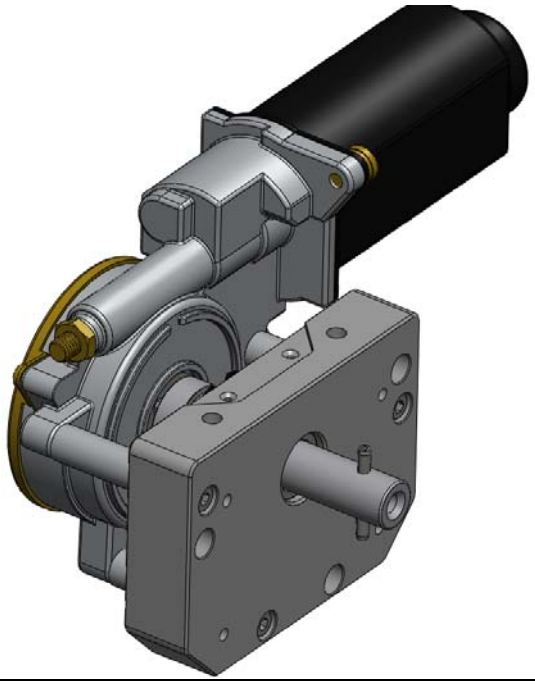
Align the mounting block so that the appropriate counter-bored holes in the block align with the screw holes in the motor casing. Using a 3 mm hex key, install the three M4 x 50 mm socket cap screws with three 3/8″ OD x 1″ round spacers to fasten the block to the motor.



Step 4: Insert roll pin in axle

Position the end of the axle so the roll pin can be inserted vertically through the side of the shaft. Using a small hammer, tap the roll pin through the hole until the pin is centered in the axle (simultaneously avoid striking fingers with hammer). It is advisable to perform this operation against a scrap wooden block to prevent possible damage to the axle.





Step 5: Assemble tire, tube, and wheel

Insert the pneumatic tube into the tire. It may help to slightly inflate the tube so it holds its shape in the tire. Mount the tube and tire onto the main wheel section, making sure the valve stem projects through the cutaway in the wheel. Note: it is entirely personal preference whether the valve stem extends toward the inside or outside of the wheel. If there is any resistance when sliding the tire on the rim, apply a small amount of dish soap and water to the inside rim of the tire as lubrication.

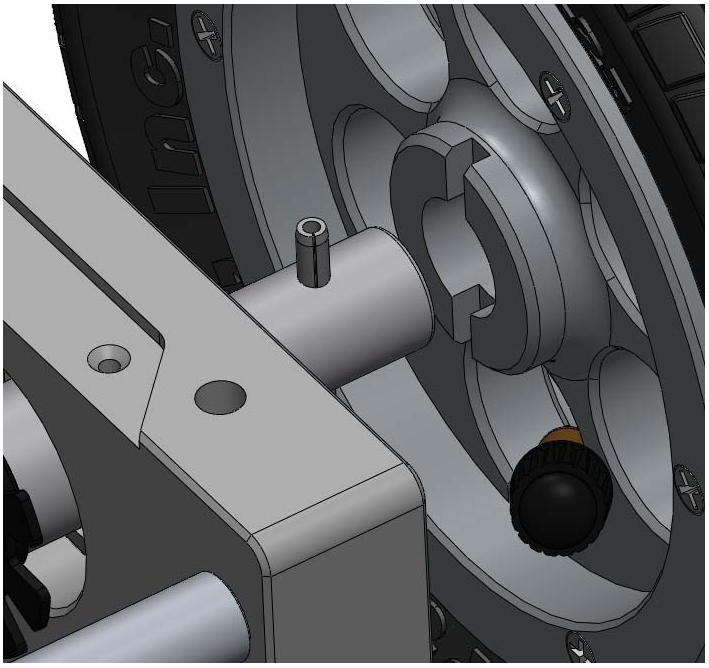
Mount the inside rim ring section using six #4-40 x ½" flat head screws as shown above. Make sure the rubber tube and tire do not get pinched when tightening down the rim. This can be accomplished by compressing the side wall of each tire while tightening the screws.

Step 6: Inflate tire to desired pressure

The recommended tire pressure is **36 PSI**; however, ideal tire pressure may vary depending on the intended terrain. For example, in more rugged outdoor environments, a lower tire pressure may improve traction and result in a smoother ride. For solid, flat ground, and for most indoor floor surfaces, a firm tire is preferable and will generally be most efficient. Experimentation is recommended to find the ideal pressure depending on robot weight and intended terrain.

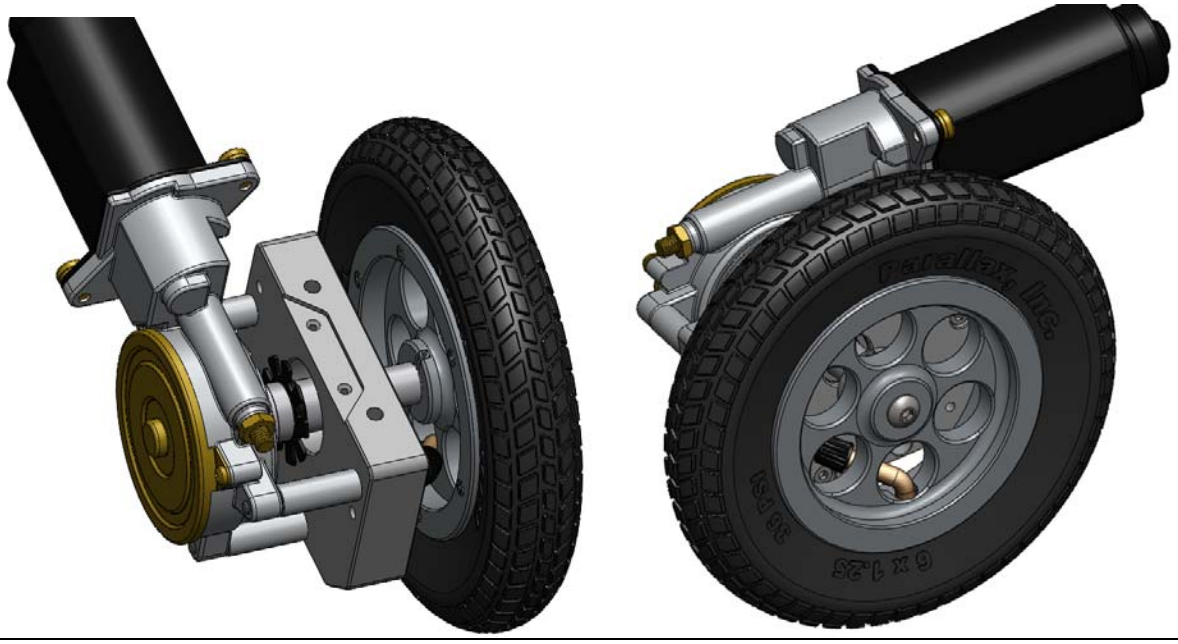
Step 7: Mount the wheel on the axle

Slide the wheel onto the axle, aligning the slot in the wheel with the roll pin in the axle as shown at right.



Fasten the wheel to the axle using the 1/4"-20 button cap screw and a 5/32" hex key.

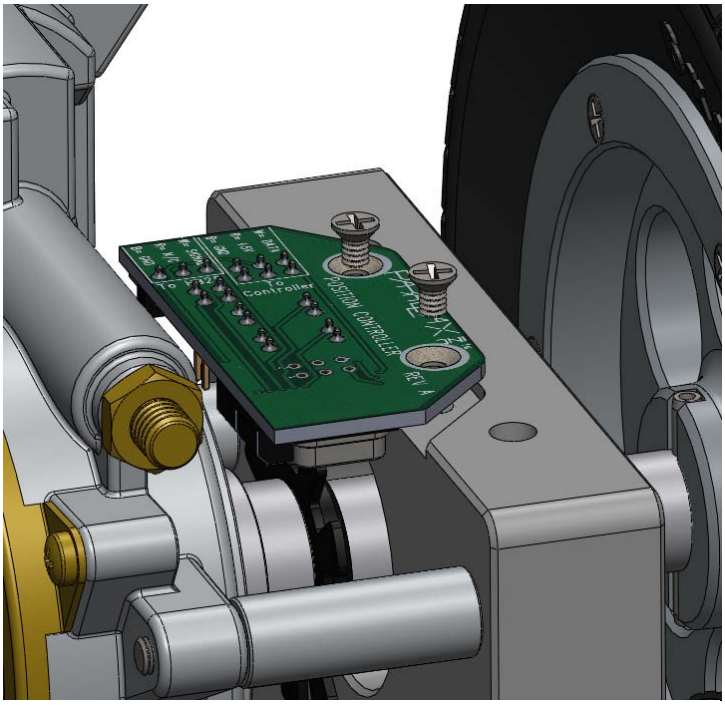
WARNING!! Do not insert fingers into the wheel holes. Do not allow hair or clothing to become entangled in the motor mount assembly.

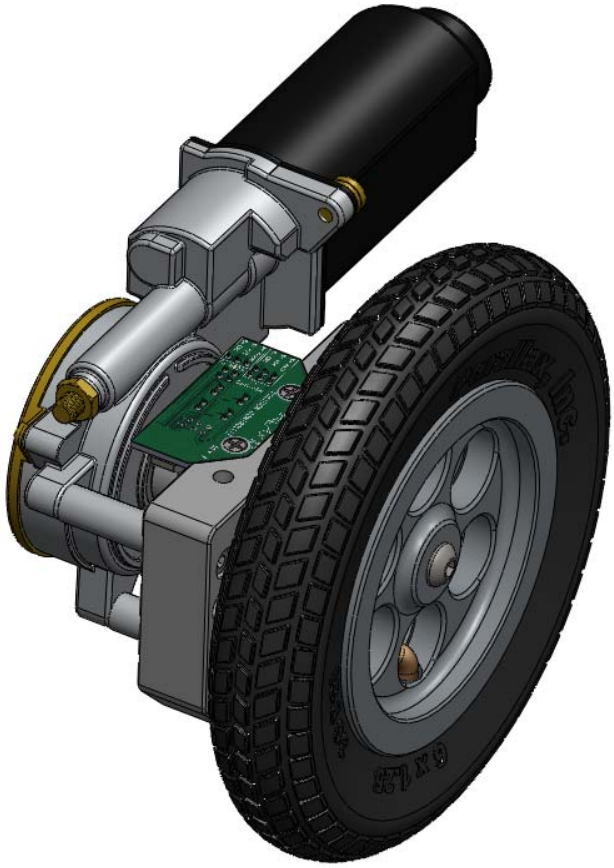


Step 8: Mount the Position Controller circuit board

Using two #4-40 x 3/16" flat head screws, mount the Position Controller circuit board in the cutout on the top side of the mounting block. Check the alignment of the plastic encoder disk to make sure it will not interfere with the optical sensors as the axle rotates.

Adjust the position of the plastic encoder disk on the axle so it passes through the sensor gap with roughly equal clearance on each side. If the encoder disk cannot be adjusted enough to avoid interference with the sensor, or if the sensors are not symmetrically aligned, it is possible to slightly re-align the sensors on the circuit board. Apply just enough finger pressure to nudge the sensor to the ideal location.



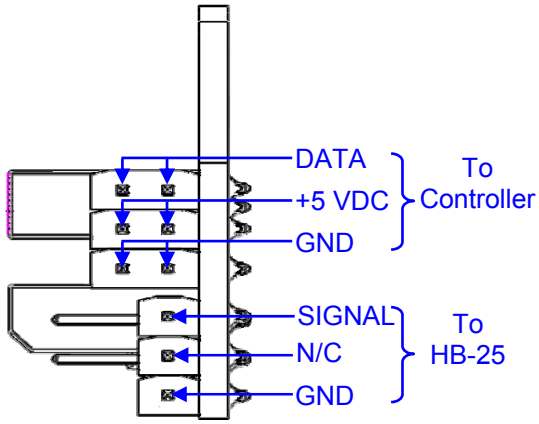


Completed Motor Mount and Wheel Kit Assembly with Position Controller

Position Controller Device Information

Pin Description

Pin Name	Function
DATA	Communication line used to send and receive data – Connect to single-wire UART bus
+5V	Regulated DC Supply Voltage (V_{CC})
GND	Common Ground
SIGNAL	Pulse output line to control an optional HB-25 motor controller



Absolute Maximum Ratings

NOTICE: These values are the limit before permanent damage may occur, but proper operation of the device is not guaranteed all the way up to these values. See the "Electrical Characteristics" section for recommended operating conditions for the Position Controller.

Parameter	Min.	Max.	Units
Operating Temperature(1)	-55	125	°C
Storage Temperature(1)	-65	150	°C
Voltage on Data Pin (with respect to Ground) (1)	-0.5	V_{CC}+0.5	V
Maximum Operating Voltage(1)	--	6.0	V

Notes:
(1) These values are taken from the Atmel ATtiny2313 device documentation.

DC Electrical Characteristics

Ambient Temperature = -40°C to 85°C

Parameter	Min.	Typ.	Max.	Units
Operating Voltage (V_{CC})	4.5	5.0	5.5(1)	V
Average Power Supply Current (I_{CC})	--	50	--	mA
Ground (GND)	--	0	--	V
Data Pin Input Low Voltage(1)(2)	-0.5	--	$0.3V_{CC}$	V
Data Pin Input High Voltage(1)(2)	$0.6V_{CC}$	--	V_{CC} + 0.5	V
Signal Pin Output Low Voltage(1)	--	GND	0.7(3)	V
Signal Pin Output High Voltage(1)	4.2(3)	V_{CC}	--	V

Notes:
(1) These values are taken from the Atmel ATtiny2313 device documentation.
(2) V_{CC} = 4.5 V to 5.5 V
(3) V_{CC} = 5 V; I_{PIN} = 20 mA

Theory of Operation

The Position Controller is designed to manage certain operations associated with wheel motion that traditionally consumed system resources, processing power, and execution time in the main microcontroller. By using the Position Controller to manage these functions instead, the main microcontroller is free to focus on more important tasks.

Each Position Controller has an on-board microcontroller which continually tracks the position and average speed of the wheel. In conjunction with the plastic encoder disk, two optical interrupter switches generate a quadrature encoded waveform which is processed by the microcontroller. Based on the rate and specific sequence of the pulses, it is possible for the Position Controller to determine how fast, and in which direction, the wheel is turning.

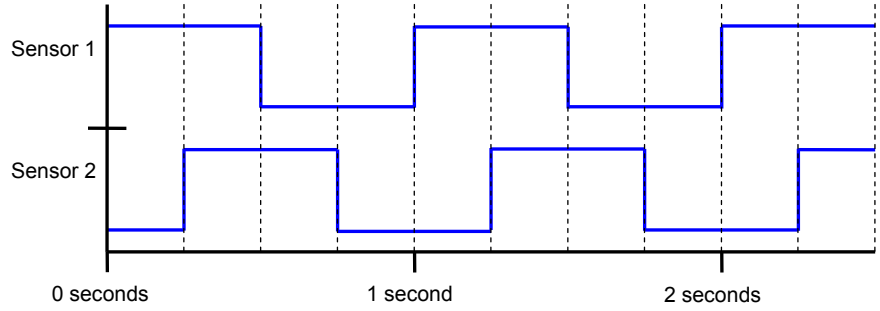


Figure 1

Quadrature encoded voltage waveforms for a wheel rotating at 4 positions per second. Notice that rising edges occur first on Sensor 1, signifying that the wheel is rotating in the positive direction.

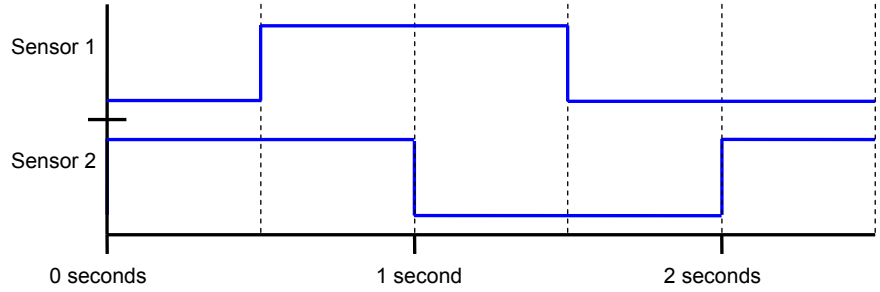


Figure 2

Quadrature encoded voltage waveforms for a wheel rotating at 2 positions per second. In this case, rising edges are seen on Sensor 2 first, signifying a negative direction of rotation.

The current position value appropriately increments or decrements by 1 each time there is a waveform transition. The current average speed value is updated every 20 ms and is an accumulated average over the previous 0.5 seconds. It is important to note that this is an average speed value, not the instantaneous speed of the wheel. See the Command Set Details section on page 12 for more information.

The Position Controller automatically generates the appropriate pulses to drive an HB-25 motor controller (#29144; sold separately). See Interfacing with the HB-25 Motor Controller on page 11 for more information. This configuration dramatically increases the Position Controller's functionality since it gains control over the wheel's position and speed rather than simply measuring them. The Position Controller stores the desired location as a set point which it continually seeks. If the wheel is rotated away from the current set point, it increases power to the motor in the opposite direction in order to return back to the original position. Additionally, when the wheel is being driven on an incline or decline, the Position Controller automatically increases or decreases power to the motor to maintain the true desired speed.

The position advancement scheme is designed to move the wheel a user-set distance (TRVL command) at a user-set maximum speed (SMAX command) with smooth user-set acceleration and deceleration (SSRR command) along the way. Each variable has a default value when powered on, but can be directly modified with specific commands (see the Command Set Details section on page 12 for more information). A typical speed profile for traveling a certain distance is shown in Figure 3 below.

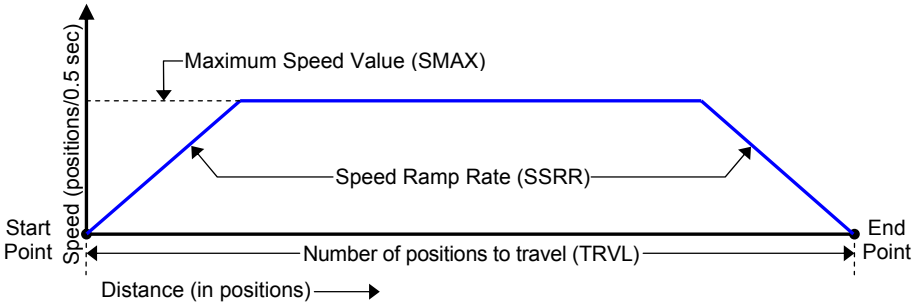


Figure 3

Typical speed profile for traveling a certain distance. The set point is advanced incrementally until it reaches the endpoint.

When the wheel advances toward the end point, it is really the set point that is being incrementally advanced. Higher speeds are achieved by advancing the set point by a greater distance each increment than for lower speeds. Since the Position Controller constantly drives the wheel to match its set point, it is able to achieve the true overall user-set speed.

The Position Controller accelerates at the user-set speed ramp rate until it reaches the user-set speed. Then it uses the current average speed to calculate the distance required to decelerate smoothly to the final end point without overshoot. When traveling in one direction, if a new position advancement value requires the wheel to reverse direction, it will still decelerate smoothly to a stop at the user-set ramp rate before re-accelerating in the opposite direction toward the new end point.

The main microcontroller communicates easily with up to four Position Controllers over a single-wire universal asynchronous serial receiver and transmitter (UART) bus. The UART operates at 19.2 kbits/sec data rate, 8-bit frame, 1 stop bit, no parity. See the Module Schematic section on page 18 for details on how the bus is electrically connected. In most cases, the data line on the Position Controller can be directly connected to the main microcontroller with no additional external components required.

Setting the Device ID

When more than one device is connected on the same bus, it is necessary to give each device a unique device ID. Each device's ID value is physically set by its corresponding jumper configuration read when powered on (see table below). Notice that the possible values are 1 through 4. Commands can address ID 0 as a special case address to send commands to all devices at once. Devices are shipped with both jumpers installed (ID value 1).

A|B	ID Value*	Binary
■■	1	001
■□	2	010
□■	3	011
□□	4	100

* ID Value 0 is reserved for addressing all devices at once.

Interfacing with the HB-25 Motor Controller

The Position Controller is designed to be interfaced with an HB-25 motor controller to control the position and speed of the motor and wheel. This can be accomplished by connecting a standard three-pin cable from the header pins marked "To HB25" to the three-pin header on the HB-25. Be sure to connect each end of the cable in the proper orientation of white "W", red "R", and black "B" as marked on both the Position Controller and the HB-25. Connect the power leads on the HB-25 as usual (positive supply to the "+" and ground to the "—").

When using the Motor Mount and Wheel Kit in the standard mirrored left motor and right motor configuration, connect the yellow lead of each motor to "M1" and the blue lead of each motor to "M2" on each respective HB-25. In this configuration, the orientation of the Position Controller on the right-hand motor must be reversed in software (see the SREV command). When wired and programmed this way, the motors will rotate in opposite directions, but the direction of rotation will be interpreted as "positive" by both Position Controllers. The result is a straight path of the vehicle (rather than rotating on center).

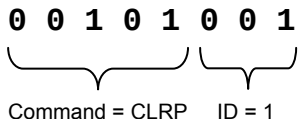
After wiring both leads identically, and setting the Position Controller on the right to be reversed orientation, both Position Controllers can be given a command to travel forward and will correctly travel in the same direction. If one of the motors starts traveling away uncontrollably (when they should otherwise be holding a position), it is likely an indication that the motor wiring is incorrect, or that the command to reverse the Position Controller's orientation was sent to the wrong Position Controller, or not sent at all.

Command Set Summary

The Position Controller's command set is grouped by category as query commands, configuration commands, and action commands. Query commands request information from a specific individual Position Controller board and generate a reply. Action commands affect the output of the Position Controller when it is being used to control an HB-25 motor controller, and do not generate a reply. Configuration commands set certain parameters in one or more Position Controller boards and also do not generate a reply. Any variables modified by configuration commands are reset to default when the device loses power.

Command codes are byte-sized with the upper 5 bits containing the command value and the lower 3 bits containing the ID value of the device for which the command is intended, minimizing the required byte count for each command.

Command Code Byte Example



Depending on the command, a command code may require zero, one, or two additional bytes of data. For commands which require two bytes of data, or commands which generate a reply, data is sent high byte first, then low byte. The Command Set Details section below contains thorough information for each command.

A Position Controller will accept a command if the ID value sent in the command code matches the Position Controller's own ID. Additionally, a Position Controller will also accept configuration commands and action commands when the ID value is 0. This is a special case ID value which allows certain commands to be sent to all devices present on the bus at once. Note that query commands do not generate a response when the special case ID value of 0 is used, since doing so would likely cause bus contention between devices.

Command Set Summary Table

	Description	Command		Hex	Binary	Single	All *
Query	Query Position	QPOS	1	0x08	00001 000	●	
	Query Speed	QSPD	2	0x10	00010 000	●	
	Check for Arrival	CHFA	3	0x18	00011 000	●	
Action	Travel Number of Positions	TRVL	4	0x20	00100 000	●	●
	Clear Position	CLRP	5	0x28	00101 000	●	●
Configuration	Set Orientation as Reversed	SREV	6	0x30	00110 000	●	●
	Set Tx Delay	STXD	7	0x38	00111 000	●	●
	Set Speed Maximum	SMAX	8	0x40	01000 000	●	●
	Set Speed Ramp Rate	SSRR	9	0x48	01001 000	●	●

* Every command can be used to address a single Position Controller device; additionally, configuration commands and action commands can be used to address all devices on the bus at once by using the special case ID value 0.

Note that the binary value for each command is simply the command number shifted left by 3 places. It is usually most expedient to store the pre-shifted binary/hex command values in a table of constants. Generating the correct command code byte can be accomplished by summing or logically OR-ing the pre-shifted command constant with the ID value of the desired device. Examples are shown in the Example Code section.

Command Set Details

QPOS – Query Position

Command #	Hex Value	Binary Value	Command Category/Type	
1	0x08	00001 000	Query	Single

Transmit Data Format
[Command:Address]

Receive Data Format
[ValueH],[ValueL]

Description: Returns the current position as a 16-bit signed value. The value is returned high byte first (ValueH) then low byte (ValueL).

The position value increments by 1 for each position traveled in the positive direction, and decrements by 1 for each position traveled in the negative direction. By default, positive direction is defined as counter-clockwise rotation when observing the wheel from the outside (screw side of the Position Controller board). When the sensor orientation is reversed (see the SREV command), the definition of positive direction is reversed.

The position value is cleared to 0 when the device is powered on. Additionally, the position value can be cleared using the Clear Position (CLRP) command. Note that positive advancement beyond +32,767 results in a rollover to -32,768.

QSPD – Query Speed

Command #	Hex Value	Binary Value	Command Category/Type	
2	0x10	00010 000	Query	Single

Transmit Data Format
[Command:Address]

Receive Data Format
[ValueH],[ValueL]

Description: Returns the current average speed in positions/0.5 sec. This is a signed 16-bit number. The current speed value is updated every 20 ms and is an average over the previous 0.5 second. The value is returned high byte first (ValueH) then low byte (ValueL).

It is important to note that this is an average speed value, not the instantaneous speed of the wheel. The difference is most noticeable when the speed changes abruptly. For example, if a speed of 10 positions/0.5 second instantly increases to 30 positions/0.5 second, the average speed value will immediately start increasing, but will take up to 0.5 second to register the new actual speed. This can be beneficial when traveling over uneven terrain since the overall average speed is usually more meaningful than a fluctuating instantaneous speed reading.

CHFA – Check for Arrival

Command #	Hex Value	Binary Value	Command Category/Type	
3	0x18	00011 000	Query	Single

Transmit Data Format
[Command:Address],[Tolerance]
{0 ≤ Tolerance ≤ 255}

Receive Data Format
[Value]
{Value = 0xFF (True); Value = 0x00 (False)}

Description: Checks to see if the device has arrived at its destination within a specified position tolerance. The position tolerance value can be 0 to 255, and must be specified in the byte directly following the command code. This function will return "True" (0xFF) if the current average speed is zero, and the current position matches the end point within ± the position tolerance value. Otherwise this function will return "False" (0x00).

TRVL – Travel Number of Positions

Command #	Hex Value	Binary Value	Command Category/Type	
4	0x20	00100 000	Action	Single, All

Transmit Data Format
[Command:Address],[NumberH],[NumberL]
{-32768 ≤ Number ≤ 32767}

Receive Data Format
N/A

Description: This command advances the end point by "Number" of positions. Number is a signed 16-bit value sent high byte first (NumberH) then low byte (NumberL). Any time the end point is moved away from the current set point, the Position Controller will accelerate at the speed ramp rate (see SSRR command) to the set maximum speed (see SMAX command) and automatically decelerate just in time to stop at the exact end point.

TRVL commands are accumulative. For example, sending a command to travel +200 positions followed immediately by another command to travel +130 positions is equivalent to sending a single command to travel +330 positions. Likewise, a command to travel +200 positions followed by another command to travel -130 positions is equivalent to a single command to travel +70 positions.

It is possible to send a TRVL command which requires the wheel to reverse its current direction of travel in order to seek the end point. When this occurs, the wheel will decelerate at the speed ramp rate in its current direction before re-accelerating in the opposite direction. This effect is also seen when a TRVL command moves the end point to a position which is still in the same direction of travel but is too close to be reached by decelerating at the speed ramp rate. Rather than stopping abruptly or decelerating faster than the speed ramp rate will allow, the wheel will decelerate smoothly past the end point to a stop before re-accelerating in the opposite direction to the current end point.

Sending a command to travel 0 positions is a special case used to bring the wheel to a smooth stop when traveling. The current speed is decelerated at the speed ramp rate and the end point is advanced to the exact location where the speed will reach zero. Note that this special case is not considered accumulative and will override the previously remaining distance to travel. However, TRVL commands sent after this will be accumulative as expected.

To immediately stop the wheel (without decelerating at the speed ramp rate) the user should send a Clear Position (see CLRP command) which effectively halts the wheel at its current position.

CLRP – Clear Position

Command #:	Hex Value:	Binary Value:	Command Category/Type:	
5	0x28	00101 000	Action	Single, All

Transmit Data Format	Receive Data Format
[Command:Address]	N/A

Description: This command resets the current position, end point, and set point back to zero. This effectively halts the wheel at its current position. The current position becomes the new starting point for position measurement and motion; and consequentially clears any TRVL commands which may be in progress.

This command can also be used as a means to "soft-reset" a Position Controller. For example, if the main microcontroller is reset while in the middle of sending a TRVL command, any data received after that will be interpreted as valid data, causing the wheel to begin traveling to an unpredictable location. By sending the CLRP command three times in a row to all present devices, it is ensured that any position advancement will be cleared; however all configuration data will remain unchanged. To completely reset a Position Controller to default including configuration data, it is recommended to cycle power. This is considered a "hard-reset".

SREV – Set Orientation as Reversed

Command #	Hex Value	Binary Value	Command Category/Type	
6	0x30	00110 000	Configuration	Single, All

Transmit Data Format
[Command:Address]

Receive Data Format
N/A

Description: Reverses the default definition for positive direction of travel. This is used when the orientation of a sensor is physically reversed. For example, in robotic applications where there is a left wheel and a right wheel, when the robot is traveling forward one wheel is traveling in the positive direction and one in the negative direction (when looking at each wheel from the side). By sending the SREV command to one of the Position Controllers, they can both be viewed as traveling in the positive direction.

The default definition for positive direction of travel is counter-clockwise rotation when observing the wheel from the outside (screw side of the Position Controller board). After receiving the SREV command, a Position Controller will interpret positive direction of travel as clockwise rotation from the same viewpoint.

This setting becomes active immediately and is persistent while the device is powered. That is, sending this command more than once does not toggle the definition for positive direction of travel; and the setting only returns to default after the device loses power.

STXD – Set Tx Delay

Command #	Hex Value	Binary Value	Command Category/Type	
7	0x38	00111 000	Configuration	Single, All

Transmit Data Format
[Command:Address],[Delay]
{0 ≤ Delay ≤ 255}

Receive Data Format
N/A

Description: Allows the user to specify the minimum delay period to wait before a Position Controller will respond to a query command and between bytes of data. Some microcontrollers like the Basic Stamp cannot receive data immediately after sending data. The Basic Stamp 2 (BS2) for example, needs at least around 330 µs to complete a SEROUT command and initialize a SERIN command before it can successfully begin receiving data. Other microcontrollers may require a longer delay or no delay at all. The minimum delay period can be calculated as follows:

*minimum delay period = (delay value * 4.34 µs) + 40 µs*

Note that the delay period calculation above is the minimum delay that the Position Controller will wait before sending responses. The actual delay may be slightly longer. The user is encouraged to experiment with different delay values until an optimal value is found for that system. The default delay value on power up is 115 which equates to about 540 µs. This delay is plenty when communicating with a Position Controller using a BS2.

Note that for query commands which return data, the data value is acquired immediately when the command is received; but is just delayed from being sent back. This may be important since the data received back could be over 2 ms old by the time the two bytes are sent using the maximum delay value of 255.

SMAX – Set Speed Maximum

Command #:	Hex Value:	Binary Value:	Command Category/Type:	
8	0x40	01000 000	Configuration	Single, All

Transmit Data Format
[Command:Address],[SpeedH],[SpeedL]
{0 ≤ Speed ≤ 65535}

Receive Data Format
N/A

Description: Sets the maximum desired rate of travel in units of positions/0.5 second (same units as the QSPD command). The speed value is an unsigned 16-bit value sent high byte first (SpeedH) then low byte (SpeedL). The default value is 36 positions/0.5 sec which equates to 120 rotations/minute. Changes in the maximum speed limit become effective immediately. It is recommended that the maximum speed value only be modified during a stopped state; however, the SMAX command technically allows for a new maximum speed to be set at any time.

The maximum speed value is considered a "true" speed because the Position Controller will maintain a constant overall average speed regardless of terrain, slope, or motor loading (as long as the driving motor has sufficient power to do so). Because the Position Controller can constantly monitor the wheel's actual speed, it can dynamically increase or decrease power to the motor in order to maintain the desired speed value. For example, if the motor is driving up a hill, the Position Controller will throttle up power to the motor to maintain constant speed; and will throttle down or even apply an opposite torque if necessary when traveling down hill.

Note that at very low speeds (around 1 position/0.5 second) motion may not be as smooth. The user may notice the individual advancements of the set point every 0.5 second; however, the overall rate of travel will still exactly match the user-specified maximum.

*While not recommended, it is possible to change the maximum speed value in mid-travel. Keep in mind that this value is the maximum speed that the position controller is allowed to travel. During motion, if the Position Controller receives a new speed maximum which is greater than the current speed maximum, it will begin accelerating at the speed ramp rate up to the new speed maximum. If the Position Controller receives a new speed maximum which is lower than the current speed maximum, the speed will instantly be clipped to the new value which will generally be abrupt and undesirable.

SSRR – Set Speed Ramp Rate

Command #:	Hex Value:	Binary Value:	Command Category/Type:	
9	0x48	01001 000	Configuration	Single, All

Transmit Data Format
[Command:Address],[Rate]
{1 ≤ Rate ≤ 255}

Receive Data Format
N/A

Description: This command sets the rate of acceleration/deceleration for the beginning/end of travel. The speed ramp rate is an 8-bit unsigned value with units of positions/0.25 sec^2. This value is the rate of change of speed (in positions/0.5 sec) per 0.5 second. The default value on power up is 15 positions/0.25 sec^2.

When a command is given to travel a number of positions, the Position Controller will begin accelerating at the speed ramp rate until it reaches the maximum speed value, or until it must begin decelerating to reach the end point without overshooting; whichever comes first.

A ramp rate of 0 is not valid since the Position Controller would never be allowed to increase its speed in order to begin traveling. Consequentially, if a speed ramp rate of 0 is received, it will automatically be increased to the minimum value of 1.

The speed ramp rate should only be modified while stopped. Attempting to modify this value while traveling will produce undesirable results.

Communication Protocol

The main microcontroller communicates easily with up to four Position Controllers over a single-wire TTL-level UART bus. The UART operates at 19.2 kbits/sec data rate, 8-bit frame, 1 stop bit, no parity. When a Position Controller is not actively occupying the bus to transmit data, it remains a high-impedance input.

Each command may require zero, one, or two additional bytes of data (see the Command Set Details section). When a certain command generates a reply, the Position Controller will wait at least the minimum transmit delay period before sending back data, as well as between bytes of data. When receiving a command, Position Controllers will wait indefinitely for the required number of bytes. This is true even when the command is not addressing the specific device's ID since the device will discard the same expected number of bytes for a command before listening for new commands.
Note that query commands should never be sent to a device ID which is not present on the bus. Since the other Position Controllers will discard the expected number of bytes based on the command, it is possible for the Position Controllers to become unsynchronized with the main microcontroller.

Figure 4 below shows an example of successful command/data exchange using the Query Speed command with a Position Controller device ID of 2.

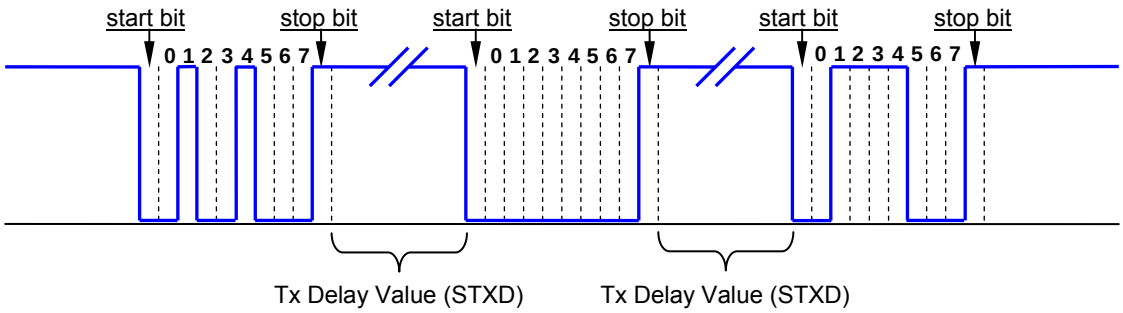
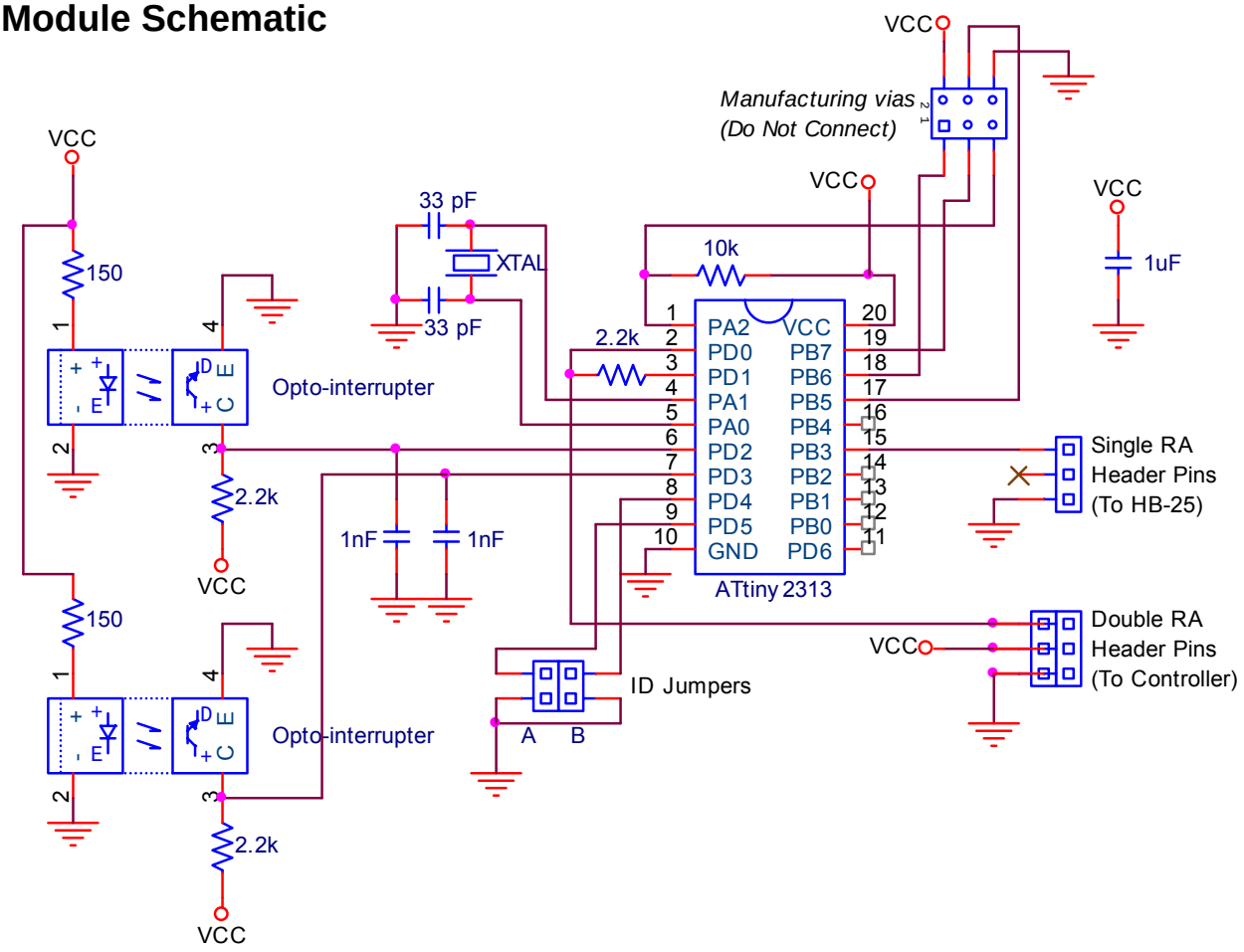
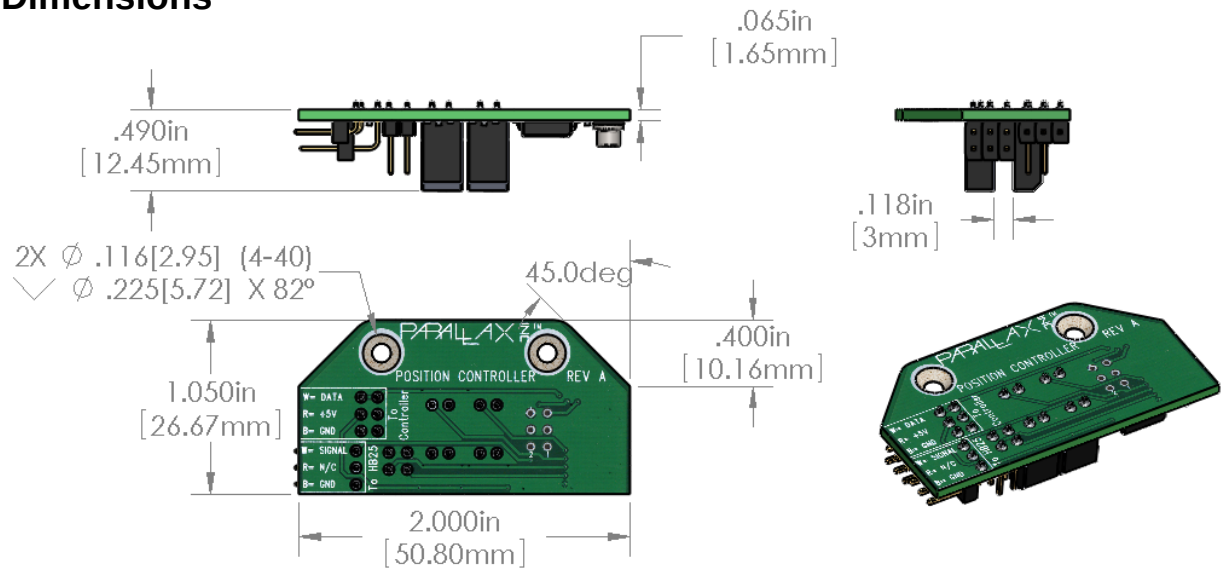


Figure 4: *Example of data exchange showing the QSPD command (0x10) addressed to ID Value 2 (0x02) and a reply of 30 positions/0.5 second (0x001E). Keep in mind that bytes are sent Least Significant Bit (LSB) First.*

Module Schematic



Dimensions

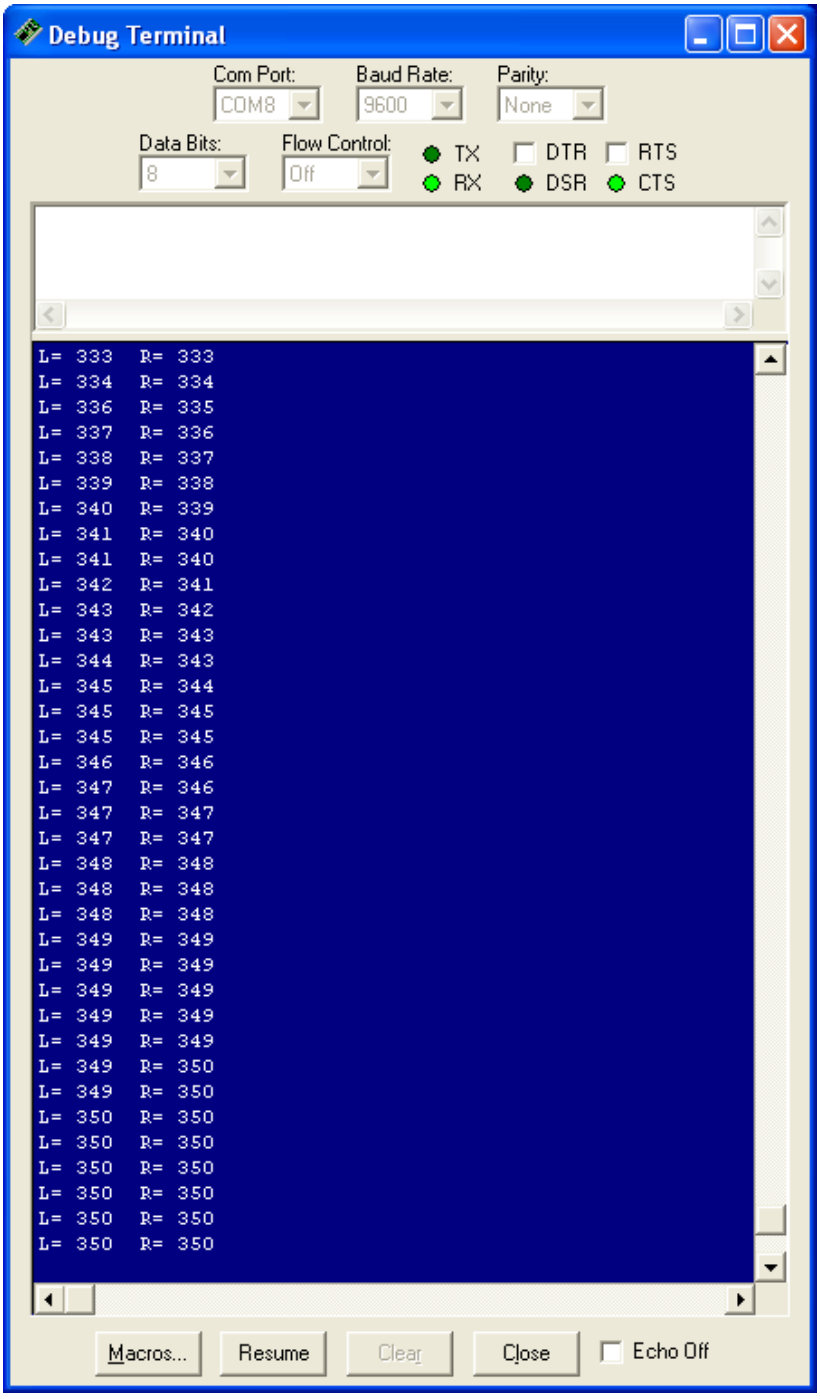


Example Code

Position Controllers are compatible with any microcontroller by following the previously stated communications protocol and instruction set. While there are many acceptable ways to execute commands in the instruction set, the following example provides convenient implementation using PBASIC on the Basic Stamp 2. The code is for a two-wheeled robot with the right and left Position Controllers physically set to ID value 1 and 2 respectively. This code is available for download from the 27971 product page at www.parallax.com.

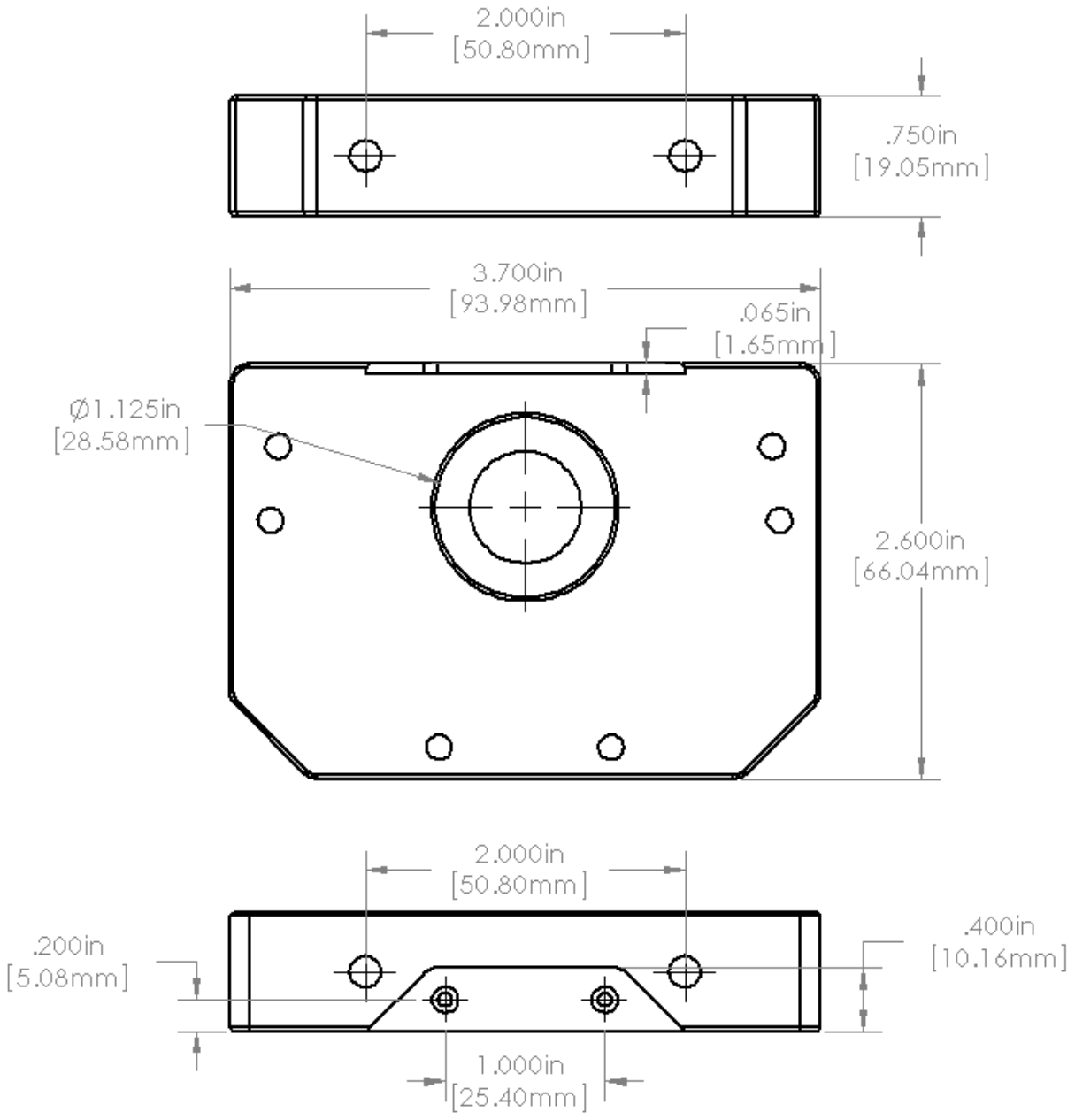
```
'-----------------------------------------------
' Position_Controller_Test_Interface.bs2
' Parallax Inc.  2008.05.02  Version 1.0
' This code is a simple test interface for a two-wheel robot design.  It sends a
' command to travel forward 350 positions and continually queries the current position
' of each wheel.
'-----------------------------------------------
' {$STAMP BS2}
' {$PBASIC 2.5}

'--- Command Value Constants ---
QPOS         CON      $08            'Query Position
QSPD         CON      $10            'Query Speed
CHFA         CON      $18            'Check for Arrival
TRVL         CON      $20            'Travel Number of Positions
CLRP         CON      $28            'Clear Position
SREV         CON      $30            'Set Orientation as Reversed
STXD         CON      $38            'Set TX Delay
SMAX         CON      $40            'Set Speed Maximum
SSRR         CON      $48            'Set Speed Ramp Rate

'--- User Constants & Variables ---
AllWheels    CON      0
RightWheel   CON      1              'ID of the right side Position Controller
LeftWheel    CON      2              'ID of the left side Position Controller
CommPin      CON      12             'communication bus pin
BaudValue    CON      32             'for 19.2kbps
Wheel        VAR      Byte           'Specifies which wheel to command for subroutines
Distance     VAR      Word           'Used to set the travel distance
RxData       VAR      Word           'Used to receive data

'--- Initialization ---
SEROUT CommPin, BaudValue, [SREV + RightWheel]  'Reverses the sensor on the right

'Go forward 350 positions
Wheel = AllWheels
Distance = 350
GOSUB GoDistance

'Continually check the position of each wheel
CheckLoop:
  Wheel = LeftWheel
  DEBUG "L= "
  GOSUB DisplayPosition
  Wheel = RightWheel
  DEBUG "  R= "
  GOSUB DisplayPosition
  DEBUG CR
  GOTO  CheckLoop
  END
```

```
'--- Subroutines ---
GoDistance:
  SEROUT CommPin, BaudValue, [TRVL + Wheel]
  SEROUT CommPin, BaudValue, [Distance.HIGHBYTE, Distance.LOWBYTE]
RETURN

DisplayPosition:
  SEROUT CommPin, BaudValue, [QPOS + Wheel]
  SERIN  CommPin, BaudValue, [RxData.HIGHBYTE, RxData.LOWBYTE]
  DEBUG SDEC RxData
RETURN
```

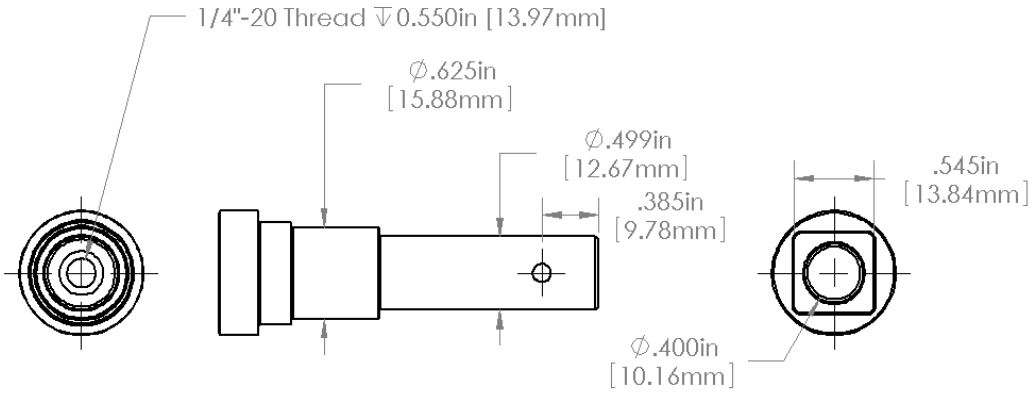


Hardware Dimensional Drawings

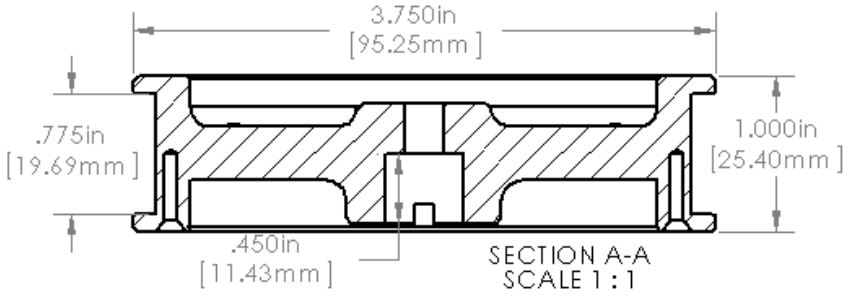
Mount Block

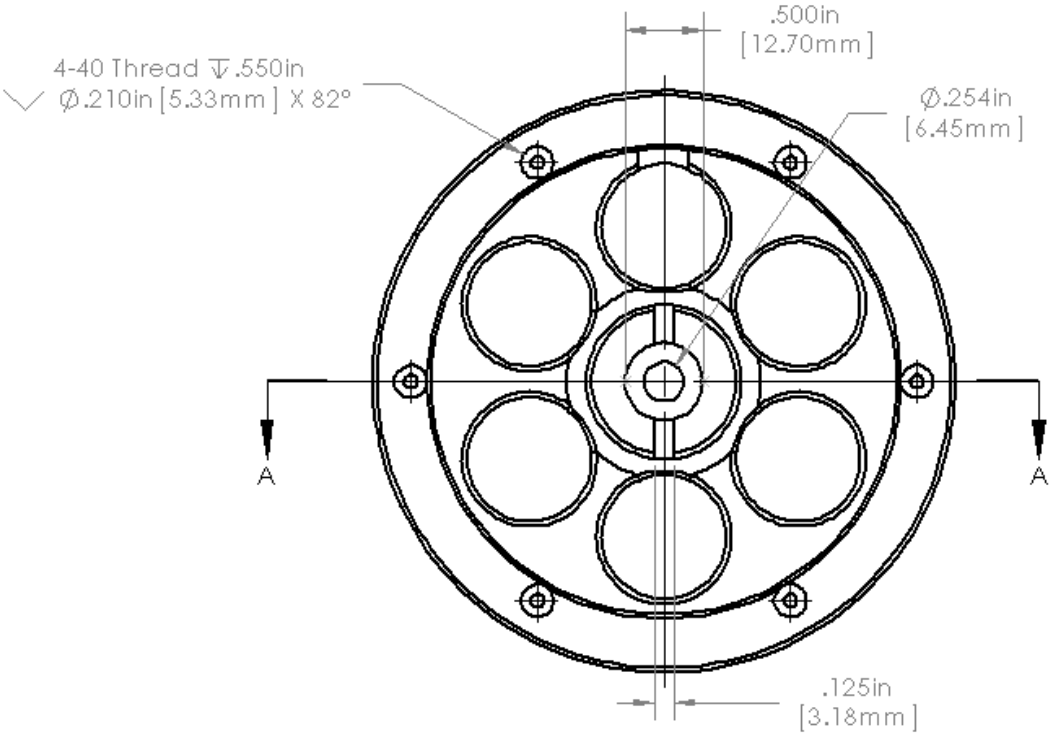


Axle


1/4"-20 Thread ↧0.550in [13.97mm]
∅.625in
[15.88mm]
∅.499in
[12.67mm]
.385in
[9.78mm]
.545in
[13.84mm]
∅.400in
[10.16mm]

Wheel


3.750in
[95.25mm]
.775in
[19.69mm]
1.000in
[25.40mm]
.450in
[11.43mm]
SECTION A-A
SCALE 1 : 1


.500in
[12.70mm]
4-40 Thread ↧.550in
∅.210in [5.33mm] X 82°
∅.254in
[6.45mm]
A
A
.125in
[3.18mm]