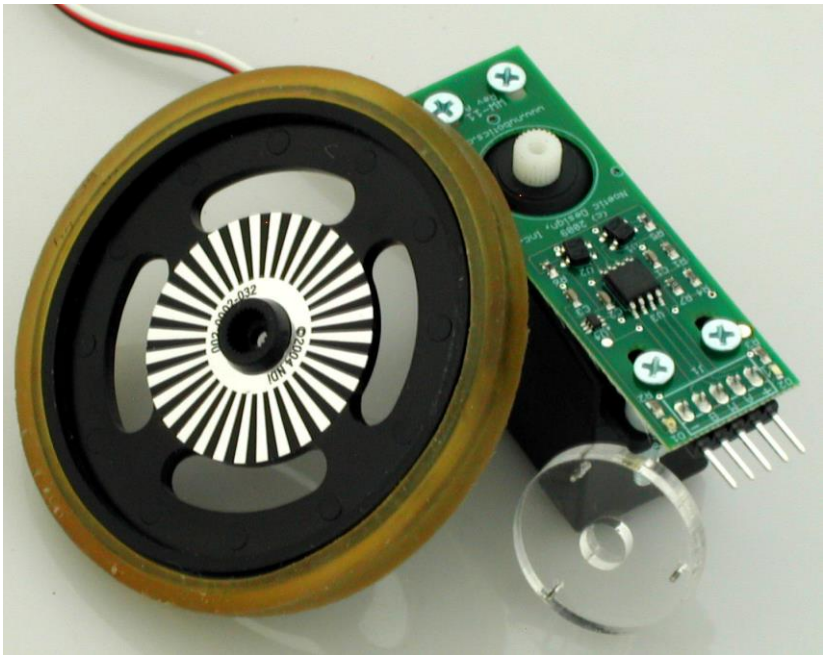
NUBOTICS

WW-11
WheelWatcher

Product Manual

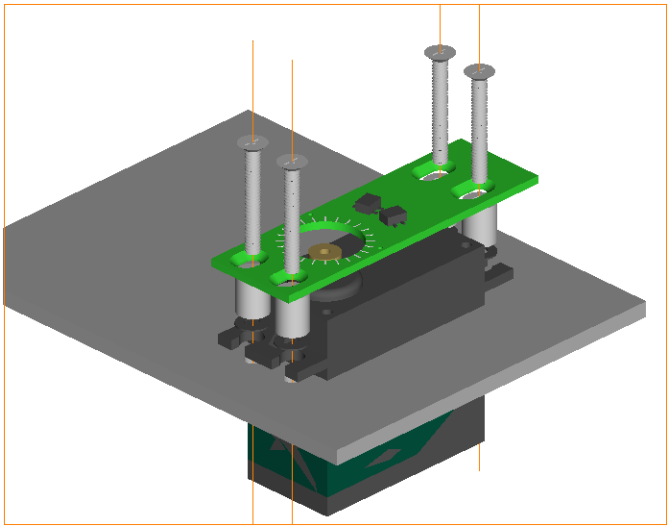


Incremental quadrature encoder system
for standard RC servos

1.03 11/28/2011

WheelWatcher Features

- easy installation
- simple interface
- preprinted 32 stripe self-adhesive reflective codewheel
- multiple interface modes for easy integration:
    - quadrature (ChA, ChB)
    - sign-magnitude (clock, direction)
    - 38400 baud serial (position, velocity)
- compatible with standard-size servos from Hitec, Futaba, and GWS
- designed for use with standard injection-molded robot wheels – any color
- code examples for common robot controllers available



Description

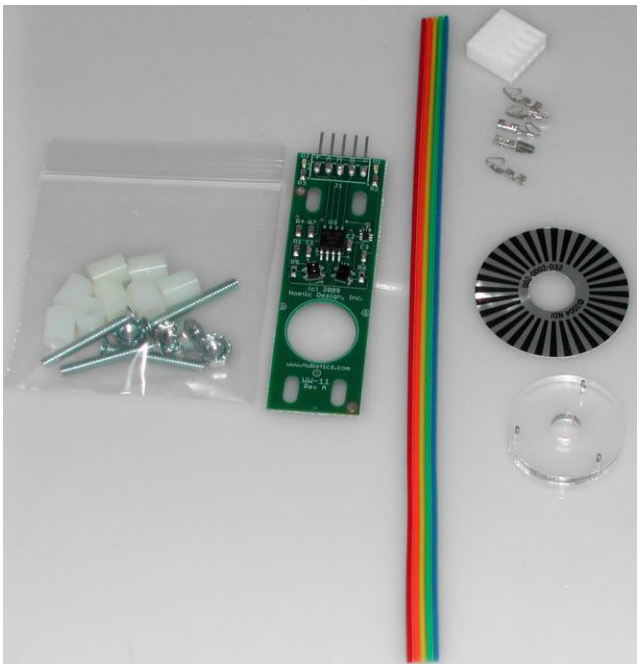
The WW-11 WheelWatcher incremental encoder system enables robot builders to quickly add closed-loop control to their robots. The WW-11 provides standard ChA / ChB raw quadrature outputs (90° phase shift between channels), decoded clock and direction signals, or serial output of position and velocity.

The clock signal toggles at each transition of ChA or ChB, providing a 4x increase in resolution compared to the number of stripes – 128 clock transitions per servo rotation – while the direction signal indicates the decoded direction of rotation, making it very easy to add to any microcontroller.

The new serial output mode does all the hard work for you – it keeps track of and reports current position (a signed 32 bit integer) and also measures and reports current velocity in encoder ticks per second. Serial data is output only when encoder ticks occur, up to a rate of 100 updates per second at 38400 baud, 8N1. Serial data is non-inverted, TTL logic level, not RS-232 (which is by definition inverted logic and +/- > 5v signal levels). See the Mode Selection section below, as well as the FAQ at the end of this document, for more details.

WW-11 Parts List

1. printed circuit board, preassembled
2. alignment tool
3. self-adhesive codewheel
4. 6" five conductor ribbon cable with matching crimp pins and housing
5. mounting hardware



Installation

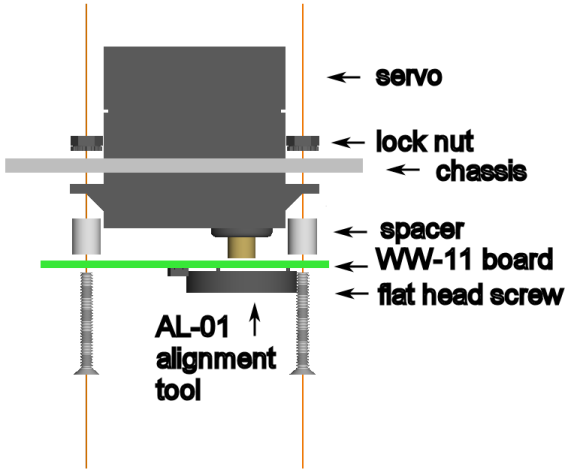
Proper spacing between the photodetectors and codewheel, as well as alignment of the photodetectors to the codewheel, is essential to proper operation.

Proper spacing depends on the board being held tightly against the top of the servo using mounting hardware, and also depends on the use of standard injection molded wheels with standard size servos. It is possible to use the WheelWatcher system with other commercial wheels, custom wheels, or even to give feedback on the rate of rotation or relative position of a leg or arm joint. However, Noetic Design, Inc. does not provide support for such custom applications.

Accurate alignment of the photodetectors to the codewheel depends on the use of the provided alignment tool to center the board's axle hole with the servo output shaft, and accurate placement of the sticker on the back of the wheel.

Installation Step by Step

PLEASE SEE TABLE 1 FOR SUGGESTED FASTENER COMBINATIONS.



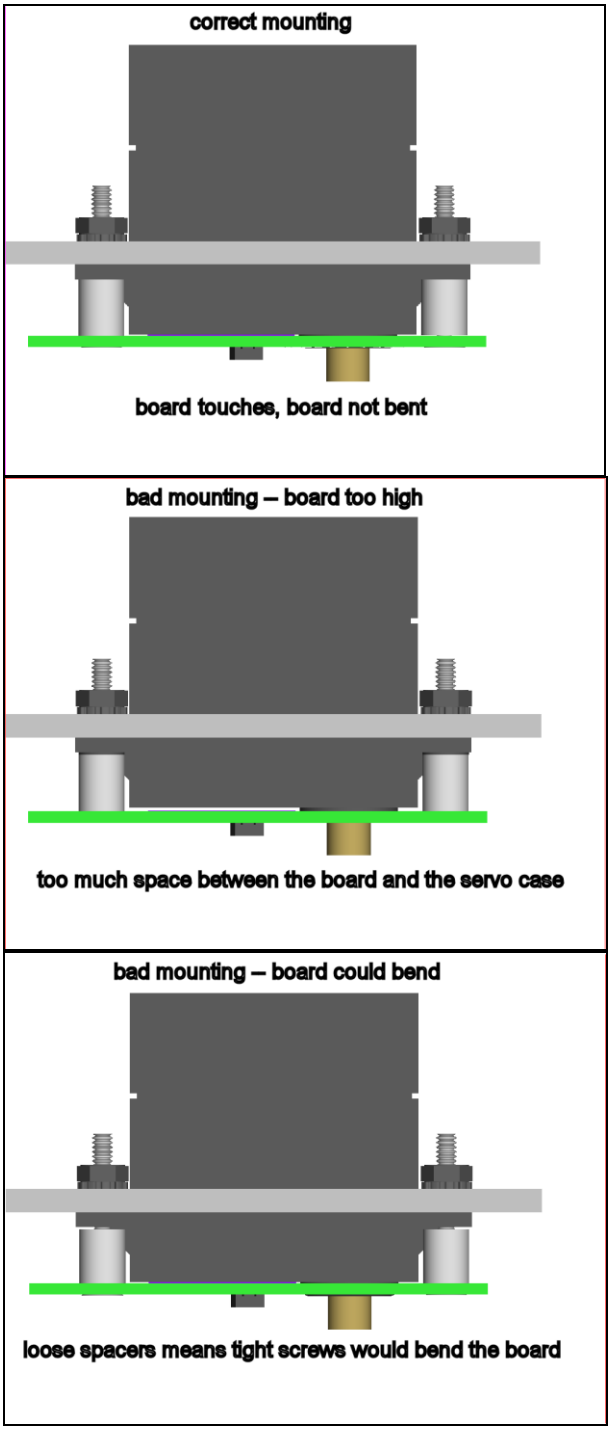
1. if the servo is already mounted on the robot, remove it[1]
2. place the servo on a solid surface (such as a table) with the output spline (the output shaft) facing up
3. line up the pins on the alignment tool with the matching holes on the board on the component (top) side
4. squeeze the board and tool together until the pins extend slightly from the far side of the board and the tool and board are parallel
5. lower the board and tool over the servo output spline, with the spline going into the hole in the center of the alignment tool, making sure the board is lined up with the sides of the servo case
6. press the board the rest of the way down onto the top of the servo; as you do so, it will slide down the pins of the tool
7. there are two different lengths of spacers provided to accommodate variation between various servo models; pick the one size that fits your servos – be sure to use all of the same length spacers; select the spacers that result in the board being held against the servo once the screws are tightened, without a gap under the board, but not so tightly that the board bends (see Table 1 for suggestions for popular servo models)
8. leave the alignment tool engaged while you mount the servo and WW-11 to the chassis, using the provided spacers, flat-head 4-40 machine screws[2], and 4-40 k-lock nuts; *remove the alignment tool once complete*
9. wire up the cable to your microcontroller; see the Connector Pinout for assembly and Interfacing Examples for details
10. place the codewheel sticker on the back of the wheel, being sure to center it with the hub; the back of the wheel is the side into which the servo motor shaft is inserted; the raised hub fits inside the sticker's center hole

Notes:

[1] if your servos are mounted to the chassis with servo tape or double stick foam tape, you might be able to leave the servos mounted

[2] You may substitute your own screws if you need shorter or longer ones; just be sure to use flat-head, 100° head-angle 4-40 screws to ensure the tops of the screws near the connector do not rub on the rim of the wheel.

Good and Bad Board Mounting Examples



Fastener Combinations For Various Servos

Table 1: fasteners

Manufacturer	Model	Short Spacer	Tall Spacer
Futaba	S3001	✓	
	S3003	✓	
	S3004	✓	
	S9001	✓	
GWS	S03N		✓
	S03T		✓
	S03TXF		✓
	S06	✓	
Hitec	HS300	✓	
	HS322HD	✓	
	HS425BB	✓	
	HS475HB		
	HS605BB	✓	
Hobbico	CS-65	✓	
	CS-67	✓	

Cable Assembly

The WW-11 kit contains a length of ribbon cable, 5 female Molex crimp pins, and a matching white Molex header. You will need to assemble this cable by performing the following steps.

1. separate about 0.75" of the ribbon cable into individual wires
2. strip off approximately 1/8" of insulation from the separated wires
3. twist the ends of each wire so that the individual fine wires are not frayed (do not connect the 5 wires to each other, just tighten each one separately)
4. notice that each crimp pin, once crimped to a wire, will grab the bare wire with one set of crimped tabs and will grab the insulated area adjacent to the bare area with another set of crimped tabs:

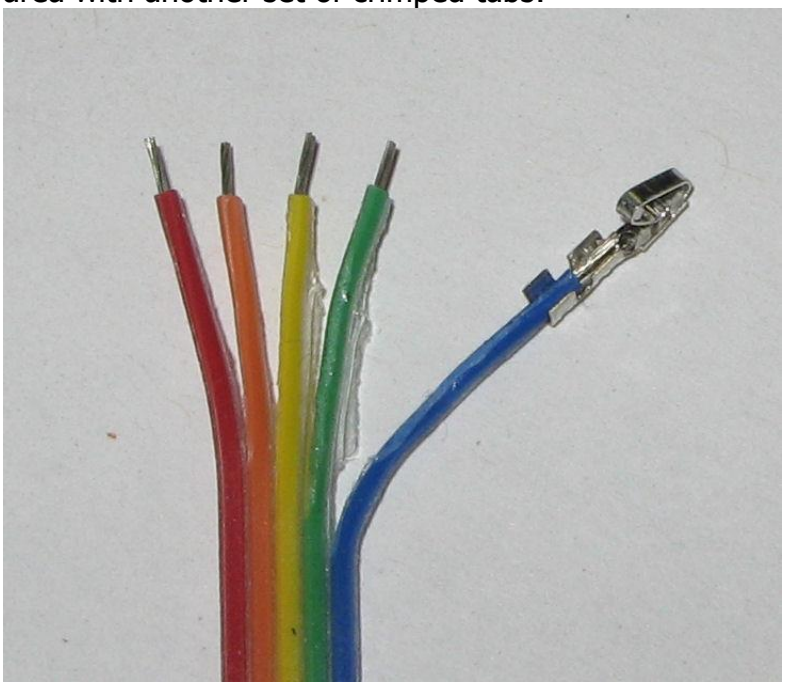


Figure 1: Ready to Crimp

5. using a crimping tool or needle nose pliers, crimp the pins one at a time to each separate wire, being sure to crimp both the insulated and uninsulated sections; it must be crimped tightly so that the pin can fit into the connector housing:

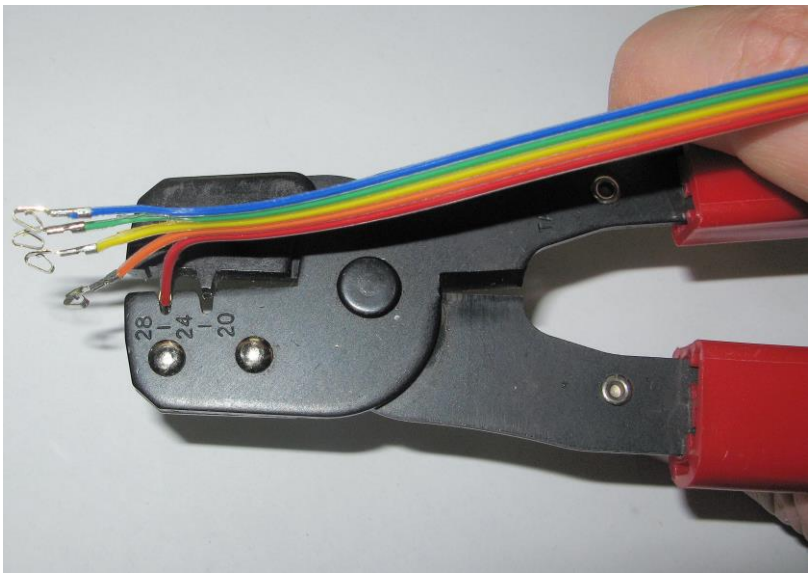


Figure 2: Crimping

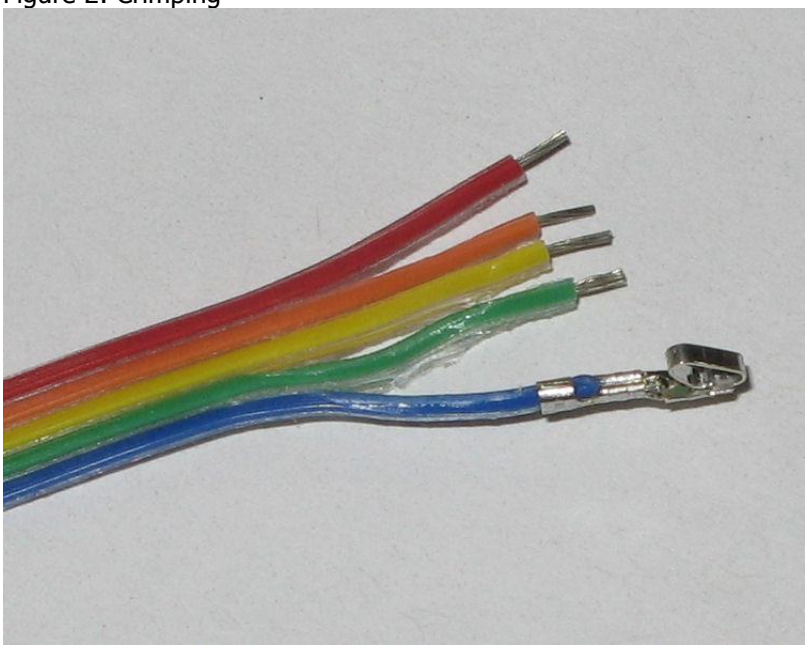


Figure 3: One Crimped

6. optional: solder the crimped bare section for a more reliable connection
7. insert the crimp pins, one at a time, into the end of the connector housing with the larger openings, such that the retention bump on the pin engages with the small opening on the side of the connector:

Figure 4: Inserting Pins

8. your cable assembly is now complete.

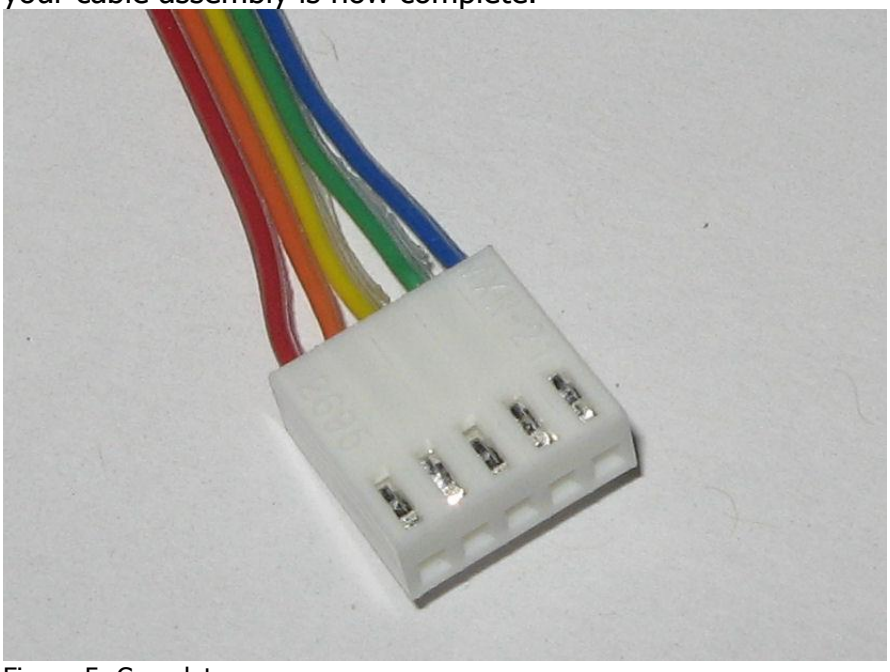


Figure 5: Complete

Connector Pinout

The color suggestions based on cable assembly photos below. We provide two different colors of ribbon cable, indicated below as Color Set A and Color Set B. Choice of color is up to the distributor.

> ***WARNING:*** ***the connector silkscreen on Rev A WW-11 boards is incorrect. It says "+ A M B –" for pins 1-5, but should read "+ B A M –" instead. In other words, the pin numbering below is correct – ignore the silkscreen.***

Table 2: pinout

Pin	Silk-screen	Quadrature Mode	Sign / Magnitude Mode	Serial Mode	Color Set A	Color Set B
1	+	Vcc			Red	Violet
2	B	ChB	Direction	/SerMode=0	Orange	Gray
3	A	ChA	Clock	TX out	Yellow	White
4	M	QuadMode=1	/SMMode=0	RX in	Green	Black
5	-	Gnd			Blue	Brown

Mode Selection

As hinted at in the connector pinout, you can select the desired operating mode of the encoder by holding specific pins high or low at reset. In firmware revisions 2 and later, this is sampled 50ms after reset (revision 1 firmware would sample immediately after reset). You can accomplish this by using a pull-up or pull-down resistor (anything up to 10K ohm would work) to either Gnd (for logic low, 0) or Vcc (for logic high, 1), depending on the mode required.

If you do not pull down any pins, the encoder defaults to quadrature outputs (ChA / ChB). These signals are square waves which are 90 degrees out of phase with respect to each other.

> ***NOTE: it can be helpful to tie pin 4 high with a resistor to Vcc to ensure it definitely goes into quadrature mode.***

If you pull down pin 4, the encoder goes into sign-magnitude mode. In this mode, the direction line is logic high when the wheel rotates clockwise, and low when it rotates counter-clockwise. The clock line toggles on every change of either sensor, resulting in an alternating pattern of 128 high and low values for each wheel rotation.

> ***NOTE: this is different from the behavior of the clock line in the WW-01 and WW-02 WheelWatcher encoders; in those older products, the clock***

line would pulse low briefly on each edge.

If you pull down pin 2 instead of pin 4, the encoder goes into serial mode. In serial mode, data will be output from the Serial TX Out pin (pin 3) at 38400 baud, 8 data bits, no parity, 1 stop bit; commands can be issued to the WW-11 using Serial RX In pin (pin 4).

Again, the TX and RX lines expect normal TTL signaling with non-inverted data. This is what one would obtain from the serial interface coming directly out of most common off the shelf microcontrollers (such as PIC or ATMEGA), the Arduino TX/RX pins (usually digital pins 0 and 1), and/or from a USB to serial converter such as the Acroname S27-USB-SERIAL (http://www.acroname.com/robotics/parts/S27-USB-SERIAL.html); see figure below.

NOTE: Do not connect the WW-11 directly to a USB to RS-232 converter nor to a standard PC serial port – the high voltages present can damage the WW-11, and the inverted signaling will result in unusable data.

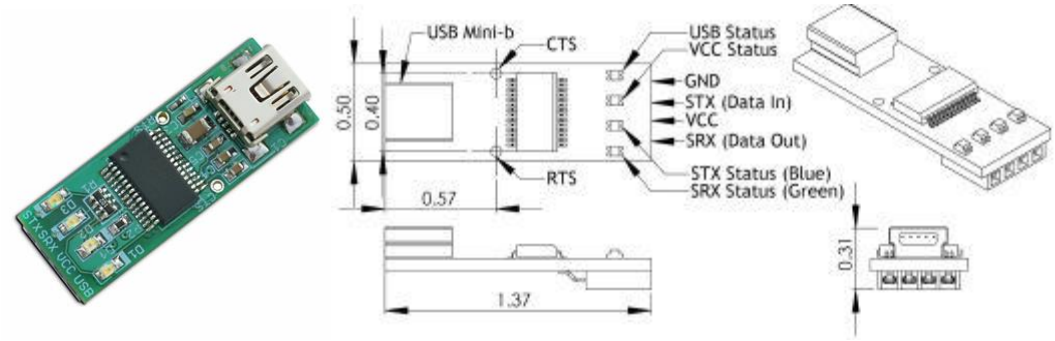


Figure 6: Acroname USB-Serial Connector

WW-12 to S27 USB-Serial Connections

Table 3: connections

WW-12 Pin	Name	S27 Pin
1	Vcc	VCC
2	/SerMode=0	10K resistor to GND
3	Tx Out	STX (Data In)
4	Rx In	SRX (Data Out)
5	Gnd	GND

Data Format

For consistency, this serial data format matches the format for our other serial-interface product, the WC-132 WheelCommander Differential Drive Controller.

After reset, the string "NWw02" is displayed, indicating the 'N'ame of the product "Ww" = WheelWatcher and the firmware revision "02."

NWw02<LF>

Once the encoder detects motion, it starts outputting distance travelled and velocity of rotation, up to 100 times per second, where distance, a signed 32 bit value, is rendered in ASCII characters with its hexadecimal equivalent requiring 8 characters; and velocity, a signed 16 bit value, is also rendered in ASCII hexadecimal, requiring 4 characters. The letter 'D' precedes the distance value, and the letter 'V' precedes the velocity value. These values are terminated with a line feed (\n = ASCII 0x0A).

D00000000<LF>
V0000<LF>

After moving 123 steps at 25 ticks per second, the output would look like this:

D0000007B<LF>
V0019<LF>

Command Set

The following commands are supported by firmware revisions 1 and 2 of the WW-11 WheelWatcher encoder when in serial mode.

D<LF> - Control Distance Output
Toggle automatic output of distance on change.

V<LF> - Control Velocity Output
Toggle automatic output of velocity on change.

E<n1><n2><LF> - Test Interface
Echo back the specified nibbles (n1 and n2) in reverse order; this is used to test the interface. The returned data will be:
E<n2><n1><LF>
For example, if you issue:
E5A<LF>
You'll receive back:
EA5<LF>

T<LF> - Request Time
This command returns the time since power on of the WW-11 in increments of 10 milliseconds (1/100 of a second resolution).
For example:
T<LF>
T00000352<LF>
Means the device has been running for 352H = 850 / 100 = 8.5 seconds. This command can be used to compensate for clock speed variations in each WW-11, which affects the accuracy of the velocity reports.

M<LF> - Request Output Mode
This command returns the current operating mode, which will always be 2.

N<LF> - Request Name and Version
This returns the current device name and firmware version (same as power up):
NWw02<LF>

. – Synchronize
If you issue a single period character, the unit will flush its buffers and issue back its own period character. This is useful to ensure communication synchronization between the unit and your controller.

Timing Diagram

The following diagram illustrates the behavior of the ChA, ChB, Direction, and Clock signals as the wheel slows down and changes direction, then speeds back up.

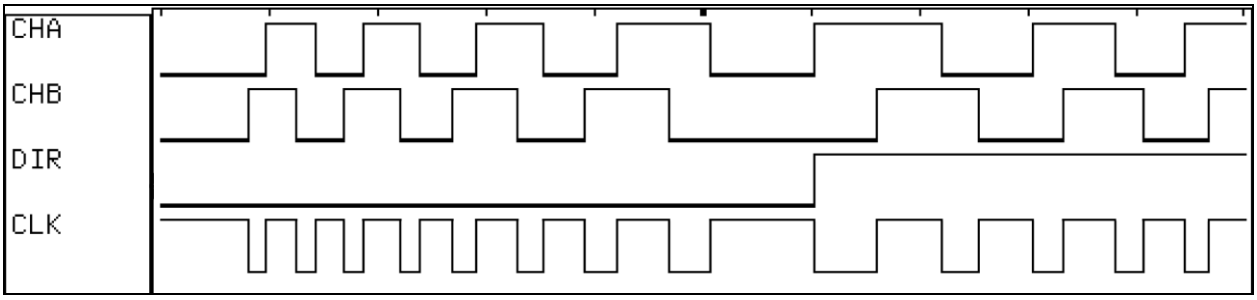


Figure 7: WW-11 Timing

Specifications

- Supply Voltage (Vcc) +3.3v to +5.5v
- Supply Current (Icc) 34mA TYP, 42mA MAX (when Vcc = 5v)
- DC Input Voltage (low) -0.5v to 0.3*Vcc
- DC Input Voltage (high) 0.6*Vcc to Vcc + 0.5v
- DC Output Voltage (low) 0.6v
- DC Output Voltage (high) 4.1v (if Vcc = 5v); 2.5v (if Vcc = 3.3v)
- DC Output Current (pins 2-4) 20mA
- Mode Sample Time 50ms after reset (firmware rev 2 and later)
- Radial misalignment TBD
- Tangential misalignment TBD
- Angular misalignment TBD
- Codewheel tilt TBD
- Phase error TBD
- Encoder Resolution Per Rev 128 (32 stripes)
- Maximum Encoder Ticks/Second TBD
- PCB center shaft hole diameter NOM 14.44mm (0.569")
- PCB width NOM 19.28mm (0.759")
- PCB length NOM 63.17mm (2.487")
- PCB thickness NOM 0.79mm (0.031")
- Detector height (see figure 10 below) 1.50mm (0.059") MIN, 1.70mm (0.067") MAX (from top to PCB)
- Detector top to codewheel (See figure 8 below) NOM 0.5mm (0.020") - 1.0mm (0.039")
MIN 0.3mm (0.012") (TBD)
MAX 1.7mm (0.067") (TBD)

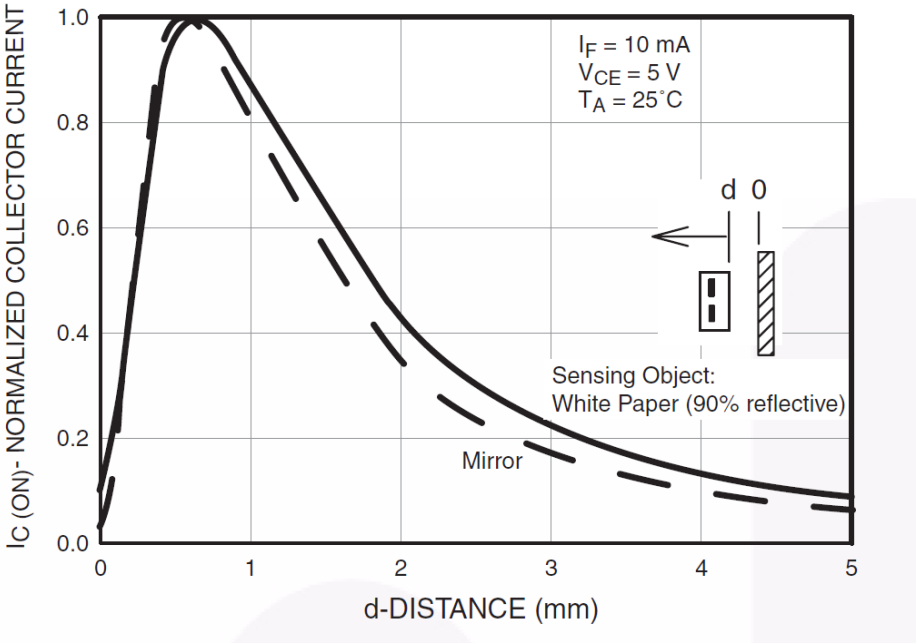


Figure 8: from datasheet for Fairchild QRE1113 Rev 1.7.0, Copyright 2011 Fairchild Corporation

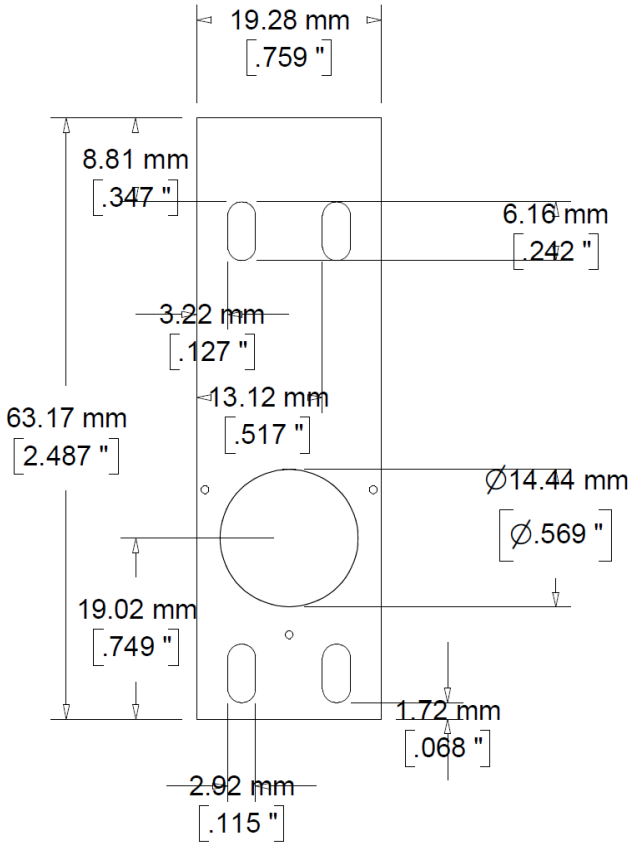


Figure 9: board outline

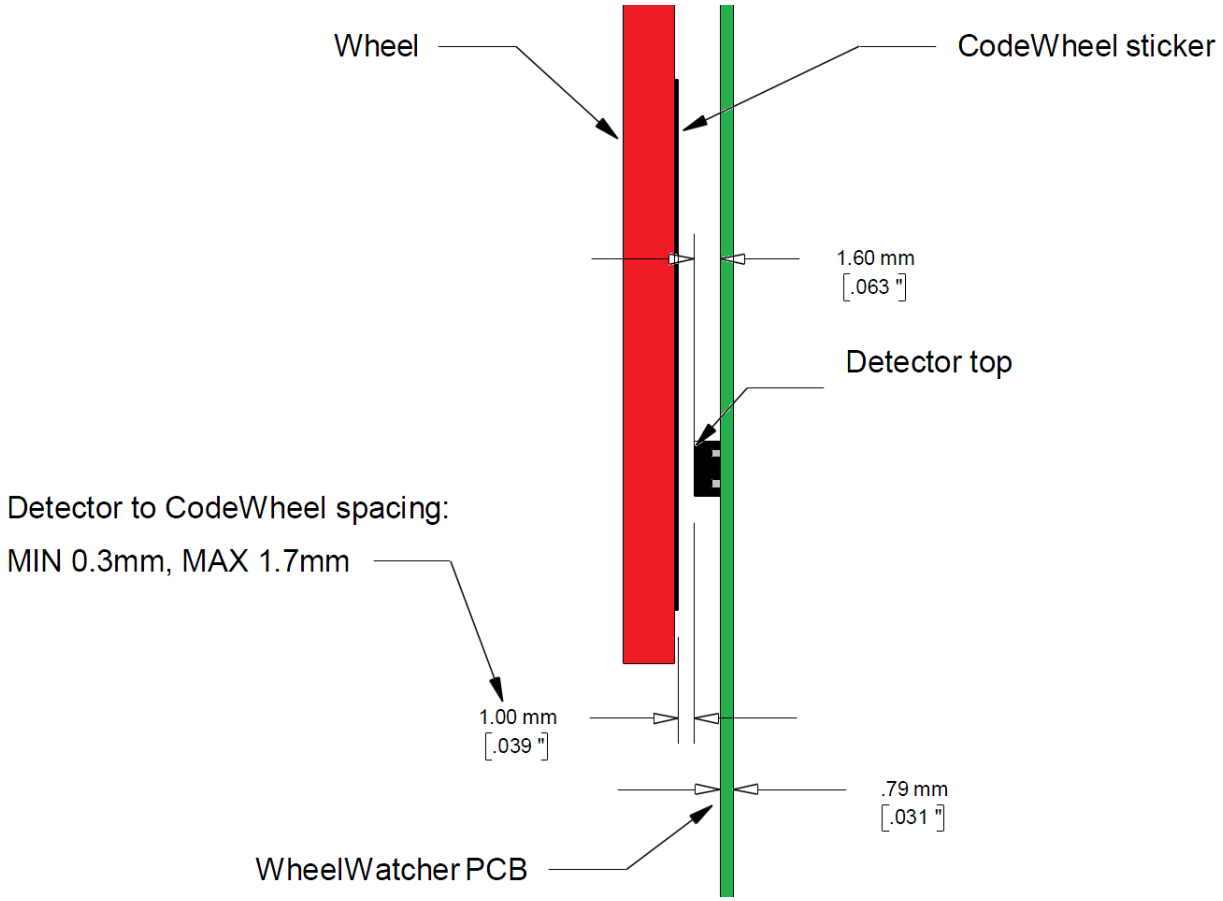


Figure 10: proper spacing

Interfacing Examples

Raw Quadrature: ChA / ChB Only

The ChA / ChB signals can be used with motor controllers that accept industrial-style incremental encoder signals, such as the Acroname BrainStem Moto, Savage Innovations OOPIC, or Solutions Cubed MiniPID.

ChA and ChB are 50% duty cycle, 90° out of phase signals, created by having two photo-detector packages spaced at a very specific angle with respect to each other, at a specific radius from the center of the axis of rotation, and expect to be used with a 32 stripe codewheel with a 50% silver/50% black radial stripe pattern.

Decoded Quadrature: Clock / Direction Only

The decoded outputs Clock and Direction are useful when interfacing the WheelWatcher to the Parallax BasicStamp II, Microchip Technologies' PIC midrange family, or other microcontrollers with hardware counter inputs or external interrupt pins. The clock line changes state (toggles) upon each transition of either ChA or ChB. The direction line is high when the wheel rotates one way, and low when it rotates the opposite way.

For example, on a Basic Stamp II, the PulsIn command operating on a Clock signal results in a direct measurement of the period of rotation of a wheel. Use that value to calculate a new servo pulse period to maintain a desired velocity.

On a PIC 16F877, tie the Clock signals to T0CKI and T1CKI, then tie the Direction signals to B4 and B5, which can issue an interrupt on change. Whenever B4 or B5 change, read the timer value, add or subtract it from a running position value based on the last direction value (since the counts in the counter were from previous motion before the direction changed), save the new direction value, then reset the hardware counter. Accurate, relatively high-bandwidth measurements of wheel velocity can be taken by using Timer2 to measure the time between edges of the Clock signals. This allows you to tightly control wheel velocity and acceleration despite the low sample rate (up to 128 clock changes per second for a 60 RPM wheel speed).

WheelWatcher™ Encoder Frequently Asked Questions

- **Can I hook a WW-11 directly to a PC serial port?**
  No. You need to provide an RS-232 level converter between the PC serial port and the WW-11, so that TTL logic levels are used and non inverted data is provided. One product that can do this is the Acroname Serial Interface Connector: http://www.acroname.com/robotics/parts/S13-SERIAL-INT-CONN.html.

- **Can I use a USB to serial adapter to interface with the WW-11, such as a Keyspan USA-19HS?**
  No. This type of USB to RS-232 dongle is not directly compatible; you can identify such an incompatible adapter by the presence of a DB-9 connector. To use such an adapter, you must add an RS-232 level converter between it and the WW-11, such as a circuit you make yourself using a Maxim MAX232 chip (http://www.maxim-ic.com/datasheet/index.mvp/id/1798) or the Acroname Serial Interface Connector. A better choice is to use a USB to serial interface that directly provides non-inverted TTL signals, such as the Acroname S27-USB-SERIAL adapter.

- **What happens if the codewheel sticker is not centered on the wheel?**
  This will result in a repeating wobbling signal pattern for each wheel rotation. It would be hard to correct for this in firmware, so go slowly and take care when mounting the sticker. This won't affect you much if you are only using the WheelWatcher for odometry (distance measurement), but will affect you if you are using it for velocity control.

- **What do the LEDs mean?**
  The YELLOW LED (D1) is connected to the Direction signal, so it turns off when the wheel rotates clockwise (CW) and on when it rotates counter clock wise (CCW).
  The GREEN LED (D2) is connected to the ChA signal, so it blinks on and off as the wheel turns.

  NOTE: *For 1 second after power up, the LEDs are set to a pattern to indicate which operating mode the WW-11 is in. If quadrature mode, yellow will be on and green off; if sign-magnitude, yellow will be off and green on; and if serial, both will be on.*

- **How can I clean the codewheel stickers?**
  We recommend only gentle cleaning with a tissue or Q-Tip moistened with clean warm water. Do not use isopropyl (rubbing) alcohol or other cleaners, as they can cloud the silver areas and render them less reflective. They could also damage the adhesive.

- **Can I run the WheelWatcher from my +6v (or +9v, ...) battery pack?**
  No. The ICs on the board require Vcc to be between +3.3v and +5.5v. We recommend the use of a regulated +5v supply.

- **How can I tell how far my robot has travelled?**
  This is called odometry. As the robot moves, simply count up edges of ChA, CB, or Clock signals. See the next question for clues as to how to convert these counts to real world distance units. Note that wheel slippage or uneven terrain will result in inaccurate distance measurements, no matter what encoder system you use; this is the bane of dead reckoning..

- **How can I measure the velocity of the wheel?**
  Two methods are commonly employed. The easiest is to count the number of clock pulses N (or changes to ChA and/or ChB) over a certain interval of time T. Velocity V = N / T in terms of counts per unit time. If T is too small you sometimes won't get any counts, even if the wheel is still turning.

  *The standard injection molded wheel used with the WheelWatcher is 2.75" in diameter (for O-ring wheels). Pi * D gives a circumference of 8.64". If you are using the Clock signal, the counts per rotation C = 128. That gives C / (PI * D) = 128/8.64 = 14.8 clocks per inch of linear travel.*

  *So, if you measured N using Clock and T in seconds, then V = (N / T) * pi * D / C in terms of inches per second.*

  The limitation with this method is that you can only update your servo control pulse width or PWM value every T seconds. For a normal servo, the highest rotation speed is usually around 60 RPM, which is one rotation per second, or 128 CLK pulses per second; at slower speeds, you will get fewer pulses per second. Your update rate (or control loop bandwidth, in control theory language) will be slow.

  Another method is to measure the time, using your microcontroller's hardware timers, between each Clock edge (or ChA or ChB edge), then take the inverse (1/T) to get counts per unit time. At 60RPM, you could update your servo control pulse value or PWM value every 8 ms instead of every second, and have much better resolution too.

  However, regardless of how well aligned the board is and well centered the sticker is, manufacturing error and alignment error will still result in some time variation from clock to clock, so we recommend that your firmware calculates a running average of the time between the 4 most recent pulses when using this method.

- **How can I control the velocity of the wheel?**
  This is an area of engineering known as control systems theory. There are many methods for using feedback to control velocity of a motor.

  In the simplest method, one measures the actual velocity of the wheel, calculates an error signal by subtracting the actual velocity from the commanded or goal velocity, multiplies that by a constant, then feeds that value to the motor. Thus, if the wheel is spinning too

slowly, the motor is told to speed up, and vice versa. What I just described is known as a P loop -- proportional control loop -- since the only error term is proportional to the difference in velocity.

One common technique that works better for varying terrain is called a PID loop, which stands for Proportional Integral Differential. Much has been written about this technique, and Google is your friend.

See the Example Code area of the Nubotics website for examples of P and PI loops using the WheelWatcher for various robot platforms.

- **Do I need to use the alignment tool?**
  Yes, it is highly recommended. If the board's sensors are not aligned well with the codewheel sticker, the timing of the various signals will be strongly affected, to the point that you may end up not getting 128 clock pulses per rotation. Further, the time between each Clock edge will not be consistent, but will instead vary from pulse to pulse, usually with a pattern that repeats every 4 clocks. Severe misalignment may result in ChA and ChB not always being in phase with each other, resulting in inconsistent counts.

- **Why am I not getting good counts from the encoders?**
  *Check the behavior of the LEDs while your servos are turning. Do this either by turning the wheels slowly by hand (rough turning can strip the gears in the servos, **so be gentle!**) with the power on to the WW-11s but off to the servos, or by using your own test program. The yellow LED should be solid on or off, and should only change state when the direction of wheel rotation changes. The green LED should blink on and off, once per stripe -- 32 times on and 32 off for each wheel rotation.*

  If the yellow LED is flashing when it shouldn't be, or if the green LED is not pulsing 32 times per rotation, make sure your wheels are on tight. If the wheels are too high above the WW-11 PCBs, there won't be enough light reflected to the sensors on the boards. It is best to use the screws provided with your servos to hold the wheels in tight.
  If you are still having problems, check the alignment of the PCB with the AL-02 alignment tool with the wheel temporarily removed. Also, check that the codewheel stickers are flat against the wheel, without bubbles under them, and that they are clean.

- **What mounting hardware is provided?**
  We provide four 4-40 x 7/8" flat head machine screws, with a 100 degree head angle so that the screw heads will not rub on the wheel. We also provide four 4-40 k-lock nuts, four .25" x .278" #4 custom spacers, and four .25" x .31" #4 spacers. See Table 1 for suggested spacers to use with various popular servo models.

- **My robot uses DC motors and hand made wheels. Can I use the WheelWatcher WW-11 with it?**
  Maybe. You will need to mount the WW-11 such that it is aligned with your motor's shaft,

and place the codewheel sticker on a flat surface (your wheel or something attached to the motor shaft) so that it is parallel to the surface of the WW-11 PCB. The distance between the top of the 2 sensors on the WW-11 and the codewheel sticker must be close to 0.8 mm (0.031") for it to function well. Beyond that, good luck. We can't support nonstandard motors, wheels, or mounting schemes -- the WW-11 is designed specifically for standard injection molded wheels and standard size RC servos.

- **I want to use the WW-11 with a Nubotics WheelCommander differential drive controller. What settings should I use?**
  Let WW-11 pin 4 float or tie it high with a pull-up resistor to select quadrature mode. For the left WW-11, connect pins 1 and 5 to WheelCommander connector J6 Vdd and ground, and pins 2 and 3 of the WW-11 to J6 ChB / ChA inputs. Connect the right WW-11 to WheelCommander connector J8 in similar manner. In the WheelCommander Setup Wizard, Motor and Encoder Settings tab, ensure that the Quadrature Settings checkbox is checked, and that the Encoder Resolution text box says 128.

- **I have used the WW-02 encoders for quite a few years and recently purchased a few WW-12 encoders. I am following the instructions and trying to use the WW-12s in sign-magnitude mode, so I am connecting pin#4 to ground, pin#1 to 3.3v, and pin#5 to ground. I am only observing 64 high and low transitions on pin#3 and not the expected 128. Is there something that I am missing?**
  The WW-11 and WW-12 use a microcontroller instead of a fixed-function quadrature decoder chip, the LSI LS7084, used in the WW-01 and WW-02. That chip's clock output generated a very short pulse at each transition of the quadrature sensors. This short pulse was problematic for some beginners to detect, so when writing the code for the WW-11/WW-12, we decided to make it easier by toggling the clock line on each transition.

  You will still get 128 transitions per rotation -- you just need to change your code to look for either edge of CLK rather than just one edge.

- **What is new about the WW-11 compared to the WW-01?**
  The WW-11 differs in these ways:
  - *5 pin, 0.1" pitch connector rather than 8 pin (2 rows of 4) 2mm connector*
  - *3 operating modes which can be selected by selectively pulling down certain connector pins:*
    - *quadrature (ChA/ChB)*
    - *sign/magnitude (CLK/DIR)*
    - *serial (38400 baud); outputs distance traveled as well as velocity*
  - *sign/magnitude mode toggles the clock line on either edge of either quadrature channel, rather than the 50us low pulse on either edge generated on the WW-01*
  - *thinner board material (0.031" rather than 0.063")*
  - *no adhesive on back of board*

Errata

WW-11 Rev A:

The connector silkscreen on Rev A WW-11 boards is incorrect. It says "+ A M B –" for pins 1-5, but should read "+ B A M –" instead.

Firmware Rev 1:

The power up states of pins 2 and 4 were sampled immediately after the microcontroller exited reset; this is too fast for slow rising power supplies. Revision 2 and later now delay 50ms after reset before sampling pins 2 and 4, ensuring proper mode selection. Noetic Design released firmware revision 2 on 6/21/2011.