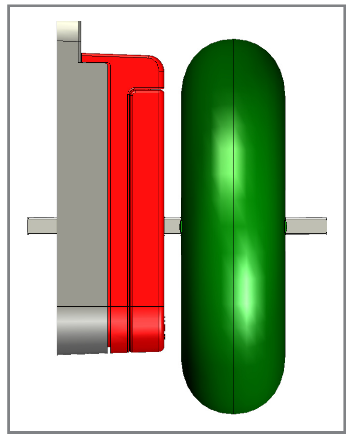
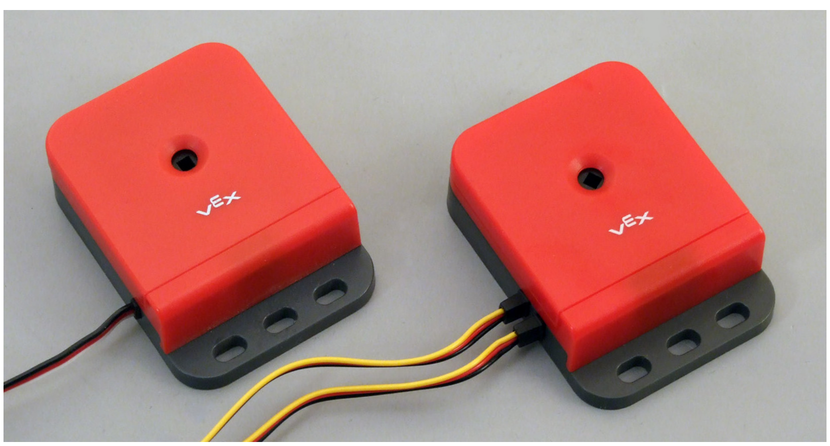
Quadrature Encoders Overview

The Quadrature Shaft Encoder detects the rotation of an axle that passes through it. It has a resolution of 360 counts per revolution (2 count intervals), and can distinguish between clockwise and counterclockwise rotation.



The Quadrature Shaft Encoder is an upgrade from the original Shaft Encoder. The original version contains only one internal sensor, which detects the slits in an internal disc as it spins, giving it a resolution of 90 counts per revolution. Only one output channel (wire) is needed to transmit the sensor data to the Vex Microcontroller.



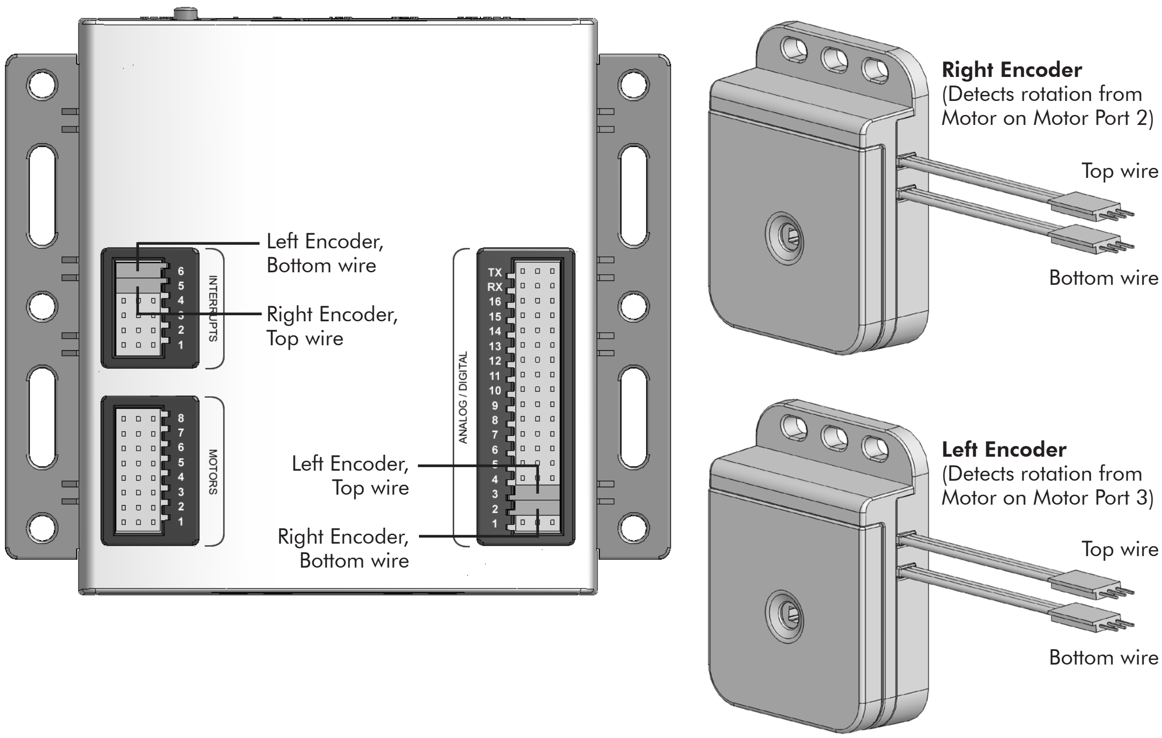
Original Shaft Encoder
Only has one output wire.

Quadrature Shaft Encoder
Has two output wires.

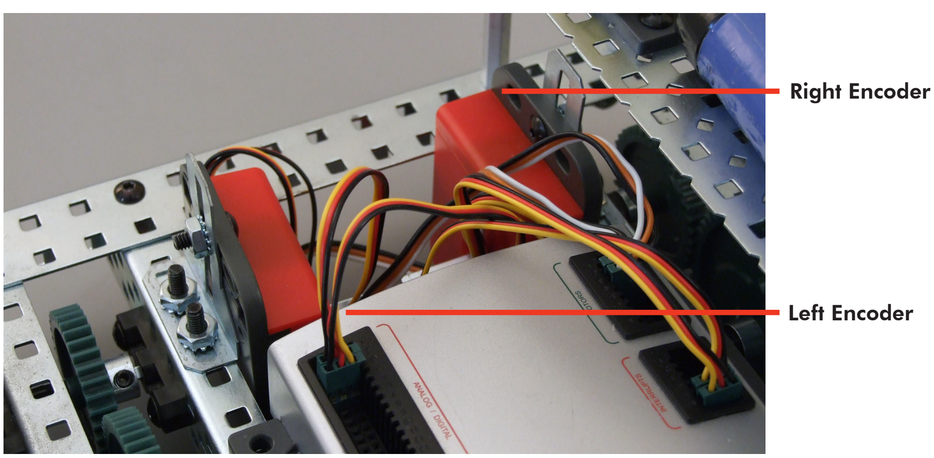
The upgraded Quadrature Shaft Encoder includes a second optical sensor which allows the sensor to detect if the internal disk is spinning clockwise or counterclockwise and increases the resolution to 360 counts per revolution (2 count intervals). Two output channels (wires) are needed to transmit its sensor data to the Vex.

Quadrature Encoders Wiring Configuration

The Quadrature Shaft Encoder is fully compatible with the existing Squarebot 2.0 and 3.0 models. Use the following wiring configuration to ensure that the Encoders count "up" when the robot drives forward, and "down" when the robot moves in reverse. Reversing the placement of the wires will cause the Encoders to count in the wrong direction (-2, -4, -6 instead of 2, 4, 6); failing to place the wires in the correct ports will result the Quadrature Encoder behaving as an original Encoder, or not at all.

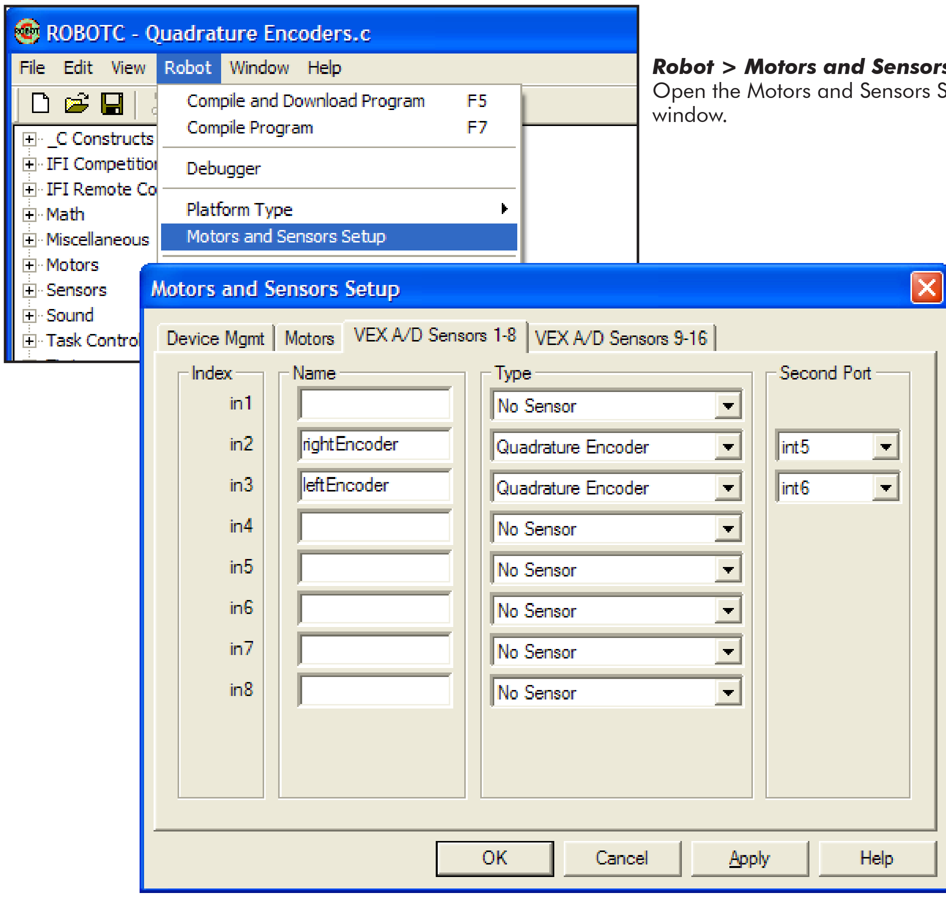


Two Quadrature Shaft Encoders mounted on Squarebot 2.0



Quadrature Encoders ROBOTC Setup

The Quadrature Shaft Encoder is also fully supported by ROBOTC for IFI (v. 1.40 and up). Use the following instructions and the wiring configuration on the previous page to correctly configure them within ROBOTC.



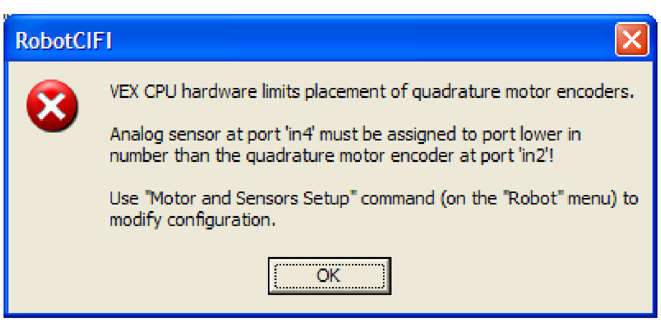
Robot > Motors and Sensors Setup
Open the Motors and Sensors Setup window.

Sensor Configuration
Select A/D Sensors 1-8.

Type “rightEncoder” next to in2, set its type as a Quadrature Encoder, and its second port as in5.

Type “leftEncoder” next to in3, set its type as a Quadrature Encoder, and set its second port as in6.

Press “OK” to complete the configuration.



Note: The Quadrature Encoders can be plugged into any of the Analog / Digital Ports (in1 through in16) and Interrupt Ports int3 through int6. However, if your robot is being configured with analog sensors (Potentiometer, Reflection, Light) as well, the Encoders must be plugged into higher port numbers for them to function.

Quadrature Encoders Sample Code

Programming with the Quadrature Shaft Encoders is straightforward, especially if you have used the original Shaft Encoders. Behaviors that involve watching a shaft as it spins forward cause the Quadrature Encoders to count up; those that involve watching a shaft as it spins in reverse cause them to count down. Here are a few pieces of sample code.

Turning with Two Quadrature Shaft Encoders

This code will cause the robot to perform a left point turn. The "rightEncoder" counts up to 270 as the motor on Port 2 spins forward. The "leftEncoder" counts down to -270 as the motor on Port 3 spins in reverse.

```
while(SensorValue[rightEncoder] < 270 || SensorValue[leftEncoder] > -270)
{
  if(SensorValue[rightEncoder] < 270) //Positive Encoder count due to
  {                                   //forward shaft rotation
    motor[port2] = 63;
  }
  else
  {
    motor[port2] = 0;
  }
  if(SensorValue[leftEncoder] > -270) //Negative Encoder count due to
  {                                   //reversed shaft rotation
    motor[port3] = -63;
  }
  else
  {
    motor[port3] = 0;
  }
}
```

Driving Straight in Reverse with Two Quadrature Shaft Encoders

This code will cause the robot to move straight in reverse for 5 full rotations (-1800 counts). The robot moves straight by monitoring the Encoder count on both sets of wheels and then making adjustments. When one side of the robot has reversed more than the other, it slows down and the other speeds up to correct the movement. When the Encoder counts are equal, the motors are set to the same power level.

```
while(SensorValue[rightEncoder] > -1800) //Reverse for 5 full rotations
{
  if(SensorValue[leftEncoder] < SensorValue[rightEncoder])
  {             //If the left side has reversed more than the right...
    motor[port3] = -50;      //slow down the left...
    motor[port2] = -63;      //and speed up the right
  }
  if(SensorValue[leftEncoder] > SensorValue[rightEncoder])
  {             //If the right side has reversed more than the left...
    motor[port3] = -63;      //speed up the left...
    motor[port2] = -50;      //and slow down the right...
  }
  if(SensorValue[leftEncoder] == SensorValue[rightEncoder])
  {             //If the left and right have reversed the same amount...
    motor[port3] = -63;      //run the motors at the same speed
    motor[port2] = -63;
  }
}
```