## **VEX Simulink Curriculum by Gary Garber is licensed under a**  [**Creative Commons Attribution-ShareAlike 4.0 International License**](http://creativecommons.org/licenses/by-sa/4.0/)**.**

# Module 2 Overview

In this module we will

* Introduce how to model the VEXnet Competition Switch.
* Create models of robot navigation based on how many rotations the motors shafts have made.
* Use sensor feedback from the VEX Integrated Encoder Modules
* Use sensor feedback from the VEX Optical Shaft Encoders

# Module 2.1 VEXnet Competition Switch

In this section we will learn how to draw a Simulink model which uses the VEXnet Competition Switch. We will build a model which can run in both TeleOp and autonomous modes.

You will learn to

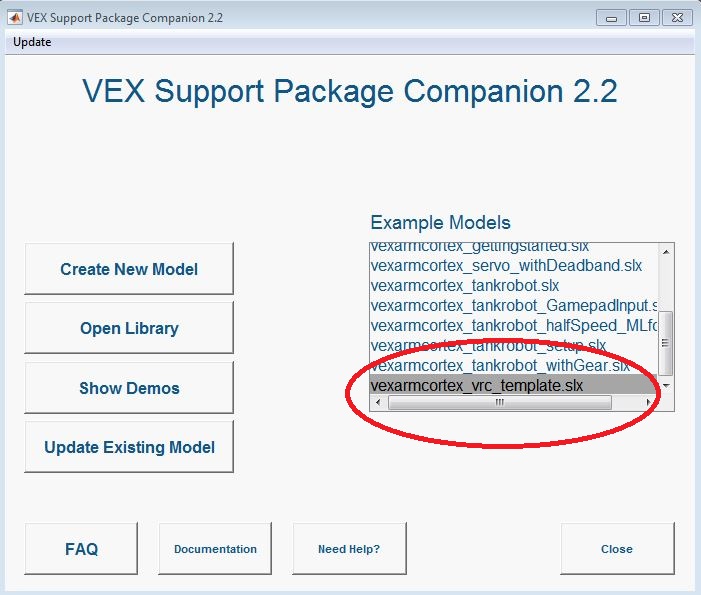
* Use the VEX Robotics Competition Template
* Create manual and autonomous control Simulink models in the template
* Run these models using the VEXnet Competition Switch hardware

The VEXnet Competition Switch is not included in your VEX EDR Starter kit. However, the VEXnet Competition Switch is essential for being able to simulate the transitions from autonomous to driver controlled modes which are used in the VEX Robotics competition. The VEXnet Competition Switch is connected to your VEX Gamepad using a standard Ethernet cable as you can see in the following [link](http://www.vexrobotics.com/276-2335.html).

When you are testing out your models, one important feature of the VEXnet competition switch is the enable/disable button. This will allow you to disable your robot. You will find this can be very important when running autonomous programs. The other important feature is the switch which transitions from an autonomous model to a driver controlled (TeleOp) model. This simulates what would happen with an actual VEX field controller.

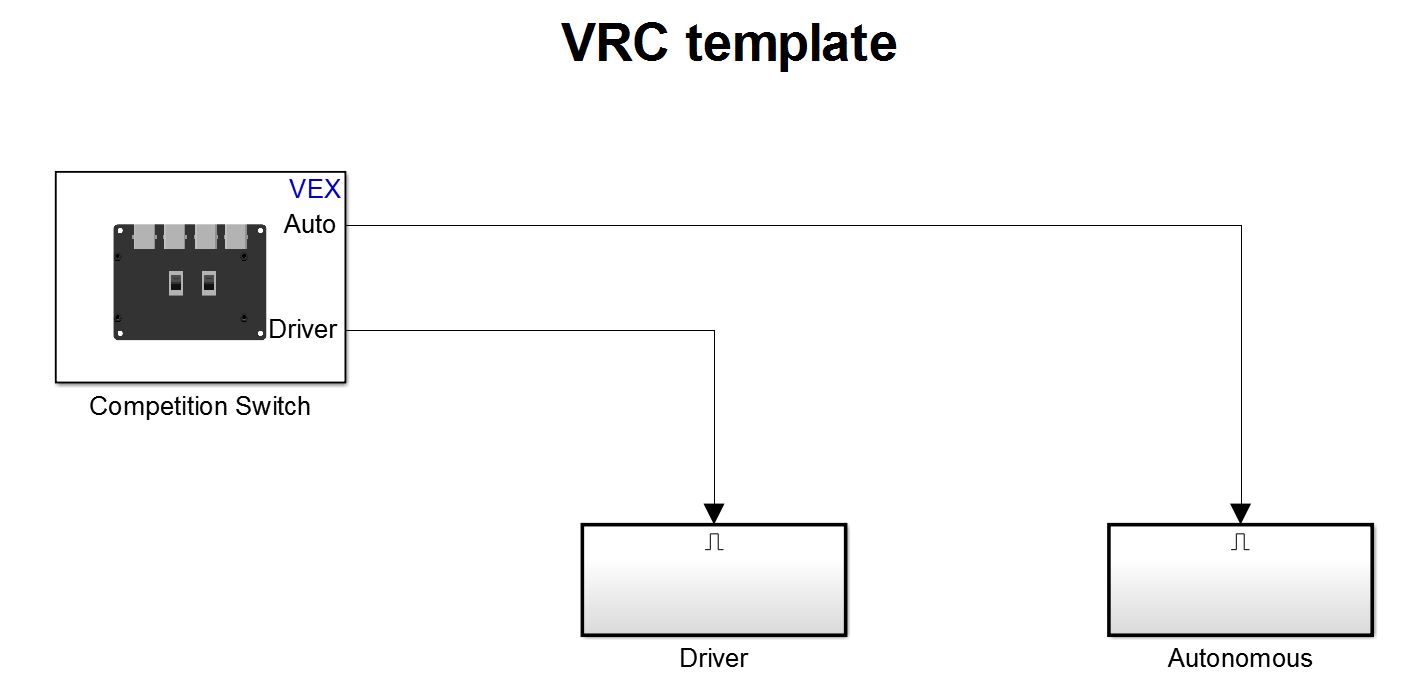
To begin,

1. Open the example model vexarmcortex\_vrc\_template.slx.
2. After you have opened the template, you should save the template under another name in your own file.



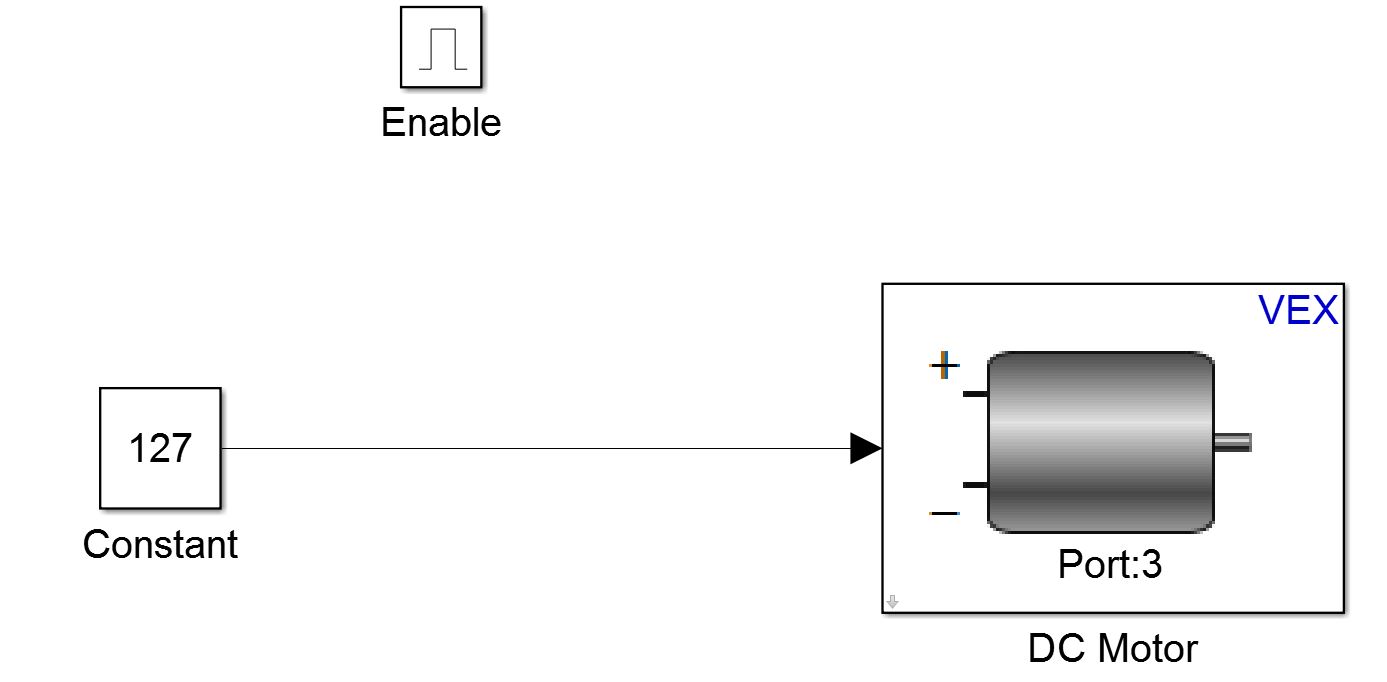
VEX Support Package Companion and choose Example Models

As you can see in the next image, when you open the template you will see a block for the Competition Switch wired to blocks for Driver and Autonomous modes. You do not need to make any changes to the top level of the Simulink diagram. Instead you can insert any models you have created directly into the Driver or Autonomous Blocks.



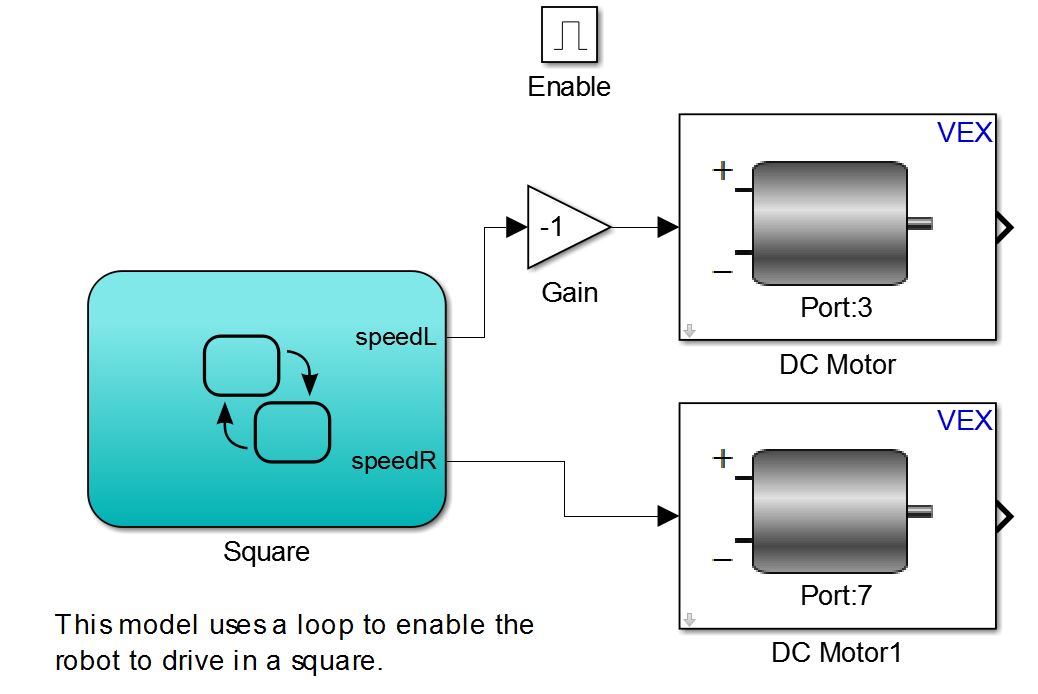
Top Level of the VRC Template

In the next image you can see, if we look inside of the Autonomous block we see a simple Simulink model which will move one motor, which is the default program. The one new block to note here is the Enable block, which allows this portion of your model to respond to the VEX Competition Switch.



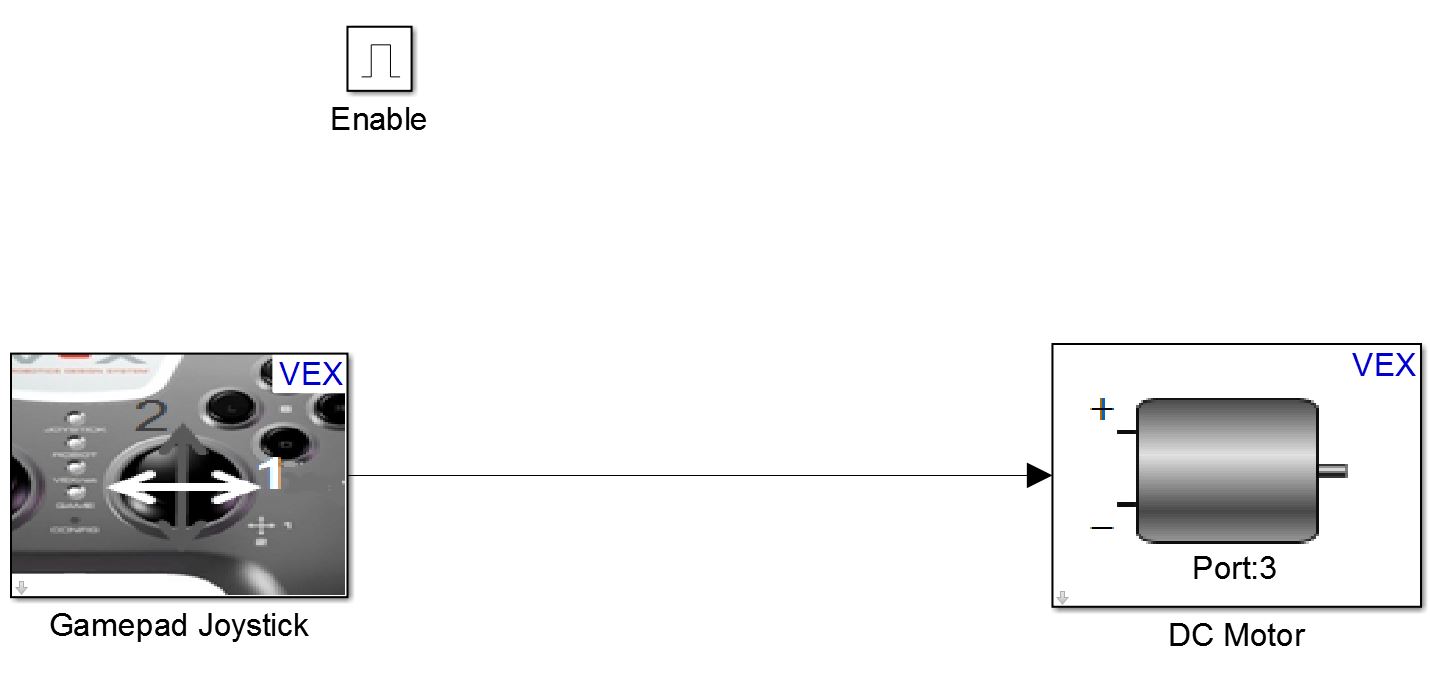
Default Autonomous Model in the VRC Template

Instead, let us replace the model which is there with our model we developed to drive our robot in a square. We can do this by cutting and pasting the blocks from our earlier model. You can now delete the default model. Do not delete the Enable block. This block enables this autonomous model when you flip the physical switch on your VEXnet Competition Switch.



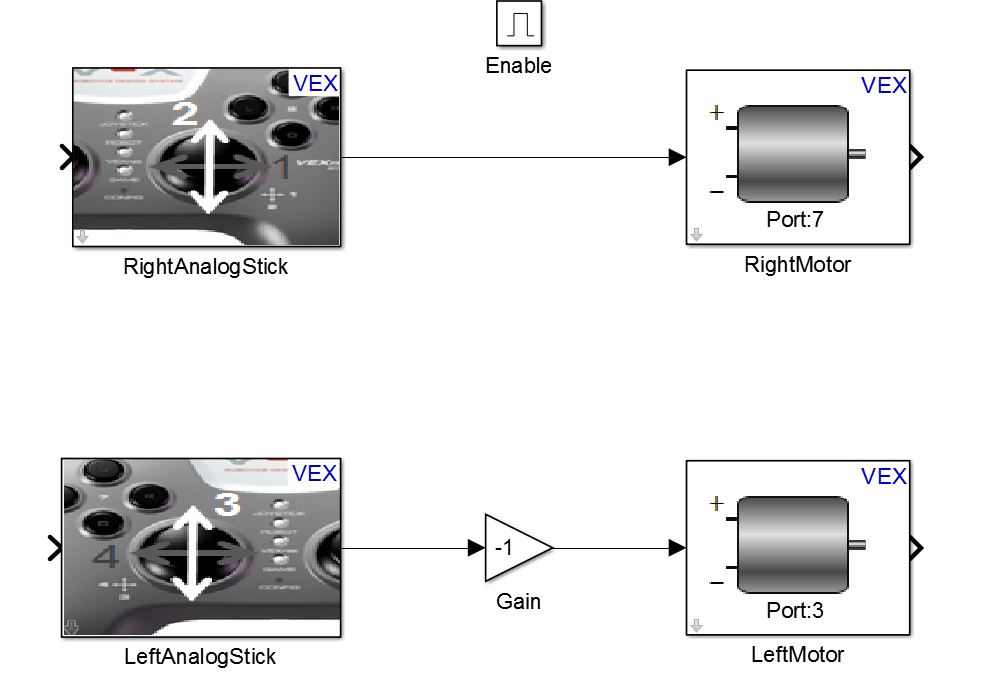
Square Autonomous Model in the VRC Template

If we look inside of the Drive block we see a default model which only controls one motor.



Default Drive Controlled Model in the VRC Template

Let us replace this with the Example model tankrobot. Remember to delete the default program but not the enable block.



TankBot Model in the VRC Template

When you deploy this model to your VEX robot you will be able to control the robot in both autonomous and TeleOp modes, by merely flipping the drive/autonomous toggle on your VEXnet Competition Switch.

In this section we have learned how to create a model using the VEXnet Competition Switch. You have learned how to import your own programs into the VEX Robotics Competition template. In the next section we will explore how to use the Integrated Encoder Module.

Additional challenge: Create a model using the template for several various combinations of the models we have created so far. For instance, combine the arcade driver model with the autonomous model which stops when the bumper sensor is pressed in.

# Module 2.2 Integrated Encoder Module

In this section we will learn how to draw a Simulink model which uses sensor feedback from the Integrated Encoder Module, which measures the number of times the motors have turned. We will write a program similar to Module 1, but instead of use time, we will use sensor feedback to transition between states in our Stateflow Diagram.

You will learn to

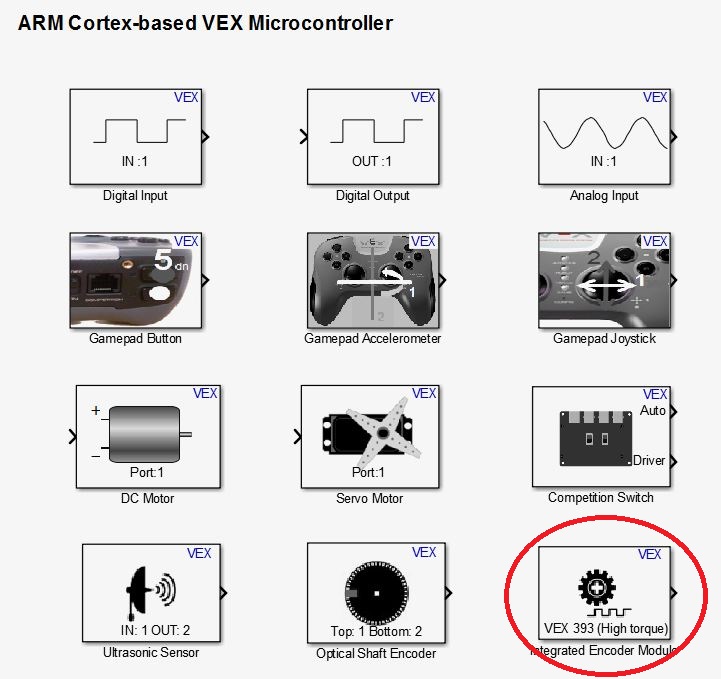
* Use the Integrated Encoder Module Block
* Write a simple model for the robot to move forward and back

The Integrated Encoder Module comes with the VEX EDR Programming Starter kit. You can also buy it separately. You will find that many teams shy away from using the Integrated Encoders because they are susceptible to resetting when exposed to static shock. This is supposedly a rare event. The encoders module attaches to the actual 393 Motor Module. You have to remove the cap from the Motor Module and replace it with the Integrated Encoder Module. A four wire cable connects the Integrated Encoder to the I2C port on your VEX Cortex controller as you can see in the following [link](http://www.vexrobotics.com/encoder-modules.html).

So although there is an actual extra physical wire, you don’t need an extra place to put this sensor since it is integrated into the motor module. Although the VEX Cortex only has one I2C port for connected the Integrated Encoder Module, you can have several of these on your robot because you can daisy chain them together.

For this robot, we have replaced only one of our physical motors on our Square Bot with a motor, which has the Integrated Encoder Module. You may want to build a robot where both of your drive motors use the Integrated Encoder Module.

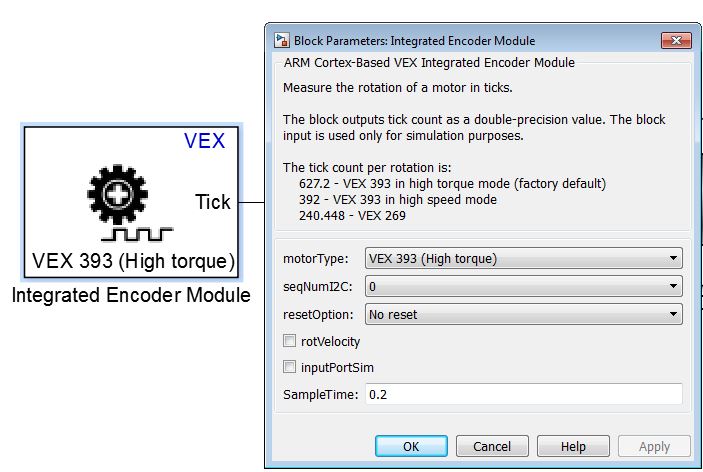
As you can see in the next image of the VEX Companion Library, you can find the Integrated Encoder Module block.



Integrated Encoder Module in VEX Companion Library

1. Drag the Integrated Encoder Module block onto your Simulink diagram.
2. Make sure the Block parameters are setup as in the following image.

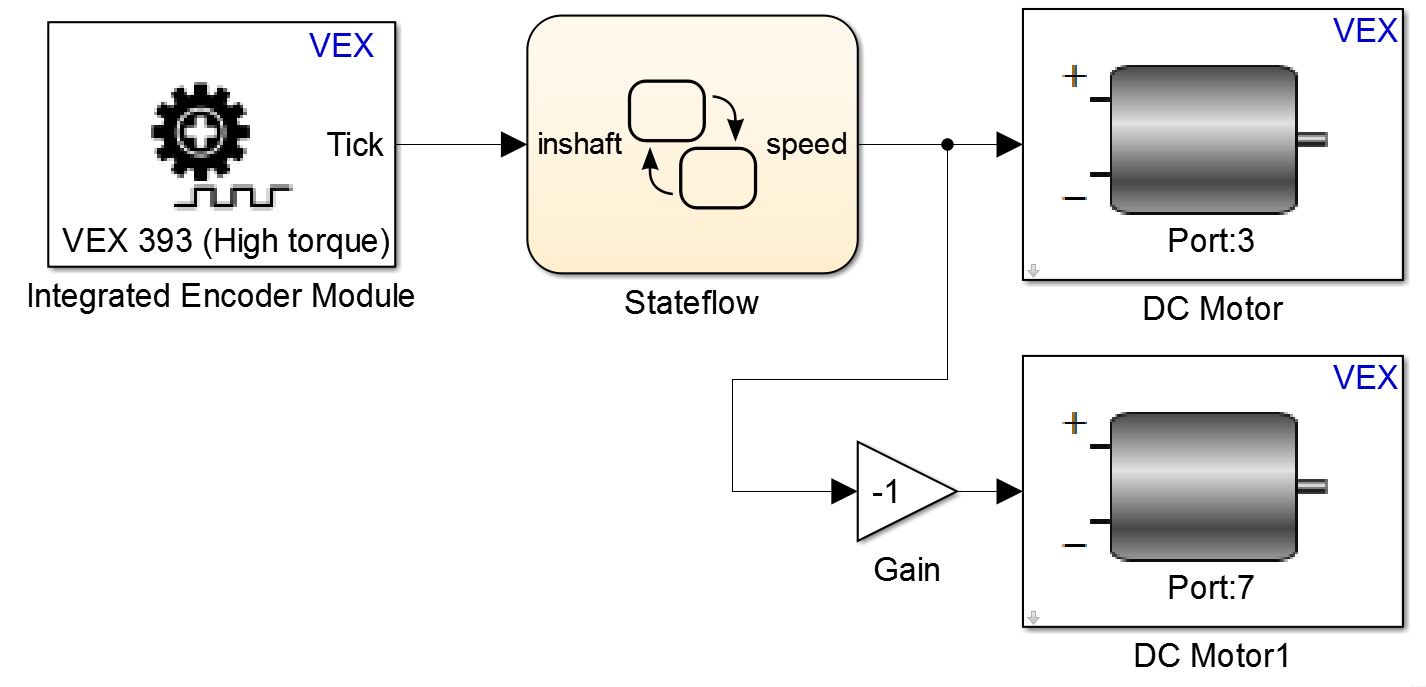
If you are using more than one Integrated Encoder Module, you would assign the order of the encoders using the Block Parameters as you can see in the next image. The seqNumI2C assigns a value to the order of the encoders.



Integrated Encoder Module Parameters

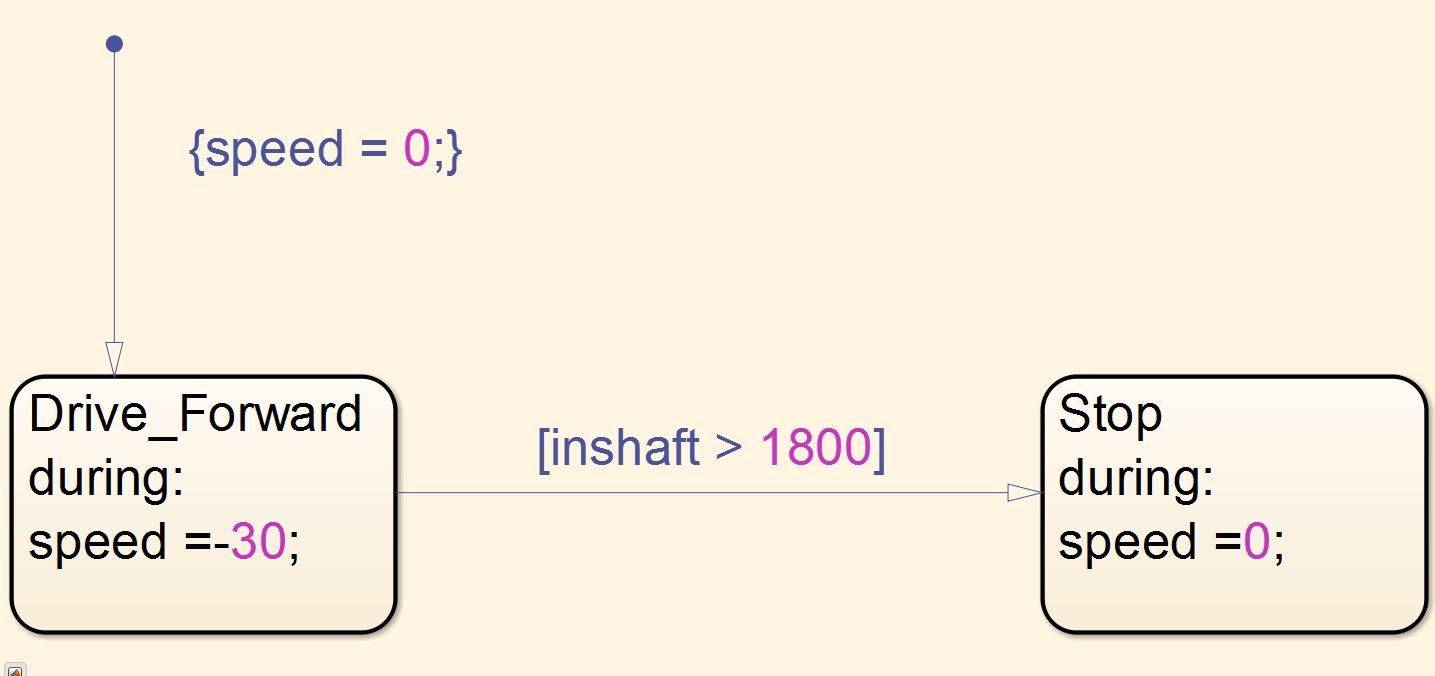
As you can see in the above image, the block parameters menu also tells us how many ticks there are per rotation of the motor shaft. This will be a crucial number in calculating how far you want your robot to move. The default setting for the shaft encoder is 627.2 ticks per rotation.

We will also create a Stateflow Chart and the motor blocks onto our Simulink diagram. The Stateflow Chart will shut the motor off when it reaches a certain Tick value. The output of the Stateflow Chart goes to the Motor Blocks. You can refer to this model in the file called IntEncoderModule.slx.



Simulink Diagram using the Integrated Encoder Module Parameters

Looking inside of the Stateflow Chart we can see the simple commands. We have created a new input port called shaft. The important thing to note here is that the speed needs to be in the opposite direction of the tick count. Our speed variable is set to -30, but we are waiting for the tick count to reach 1800.



Stateflow Chart using the Tick Count as the Condition

When you deploy this program to your VEX Robot, you will find it will rotate until the shaft encoder reaches the desired value.

In this section we have learned how to use the Integrated Encoder Module. You will find that you will have much better precision and consistency in the resulting behavior of your robot by using the Integrated Encoder Module as opposed to using time as a condition. In this next section we will use the Optical Shaft Encoder.

Additional challenge: Create a model where the robot the robot drives in a square, but instead of using time as the condition in your Stateflow Chart, use the tick value of the Integrated Encoder Module.

# Module 2.3 Optical Shaft Encoder

In this section we will learn how to draw a Simulink model which uses sensor feedback from the Optical Shaft Encoder, which measures the number of times a shaft has turned. We will write a program similar to the last section, but instead of using the Integrated Encoder Module block, we will use the Optical Shaft Encoder Block.

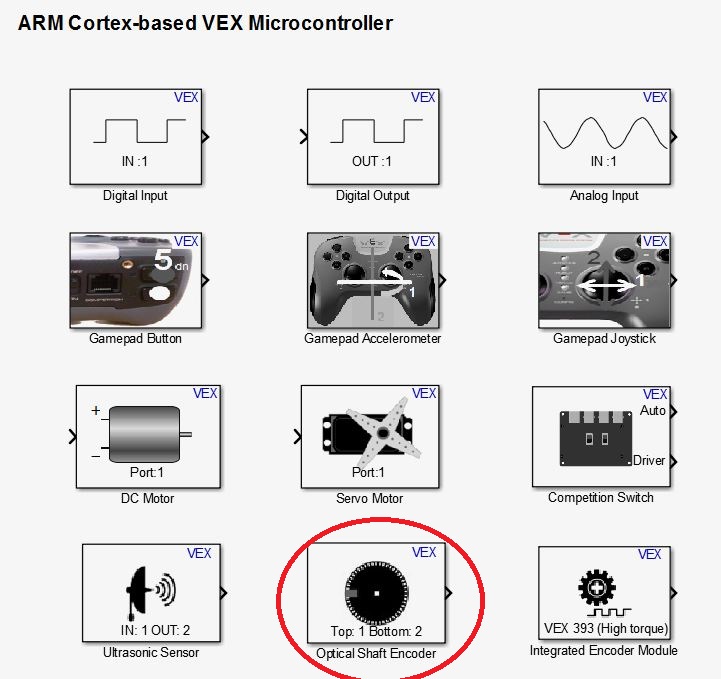
You will learn to

* Use the Optical Shaft Encoder Block
* Write a simple model for the robot to move forward and back

The Optical Shaft Encoder is a separate sensor. The Optical Shaft Encoder is much larger and bulkier than the Integrated Encoder Module. However, the Optical Shaft Encoder is not susceptible to static charge shock in the same way the Integrated Encoder Module is. Additionally, the Optical Shaft Encoder can be placed anywhere in your robot. This makes the Optical Shaft Encoder ideal for usage in the final stage of an arm/appendage so you know how far a shaft has turned. More info on the encoder is available at the following [link](http://www.vexrobotics.com/vexedr/products/accessories/electronics/276-2156.html).

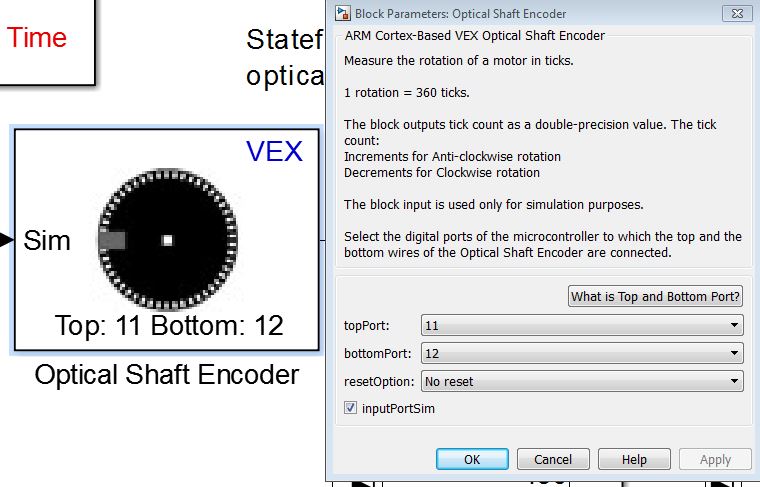
The Optical Shaft Encoder is connected to the VEX Cortex by a normal three wire cables. However, it is worth noting that the Optical Shaft Encoder uses two output cables. This allows the Optical Shaft encoder to measure both position and direction of rotation. The older VEX shaft encoders, which only had one cable, could measure position, but you did not know which direction the shaft was turning.

1. You will need to use a longer axle on one of the wheels of our Square Bot
2. Then you need to attach the Optical Shaft Encoder to that axle.
3. From the VEX Companion Library you can find the Optical Shaft Encoder block and drag it onto your Simulink diagram.



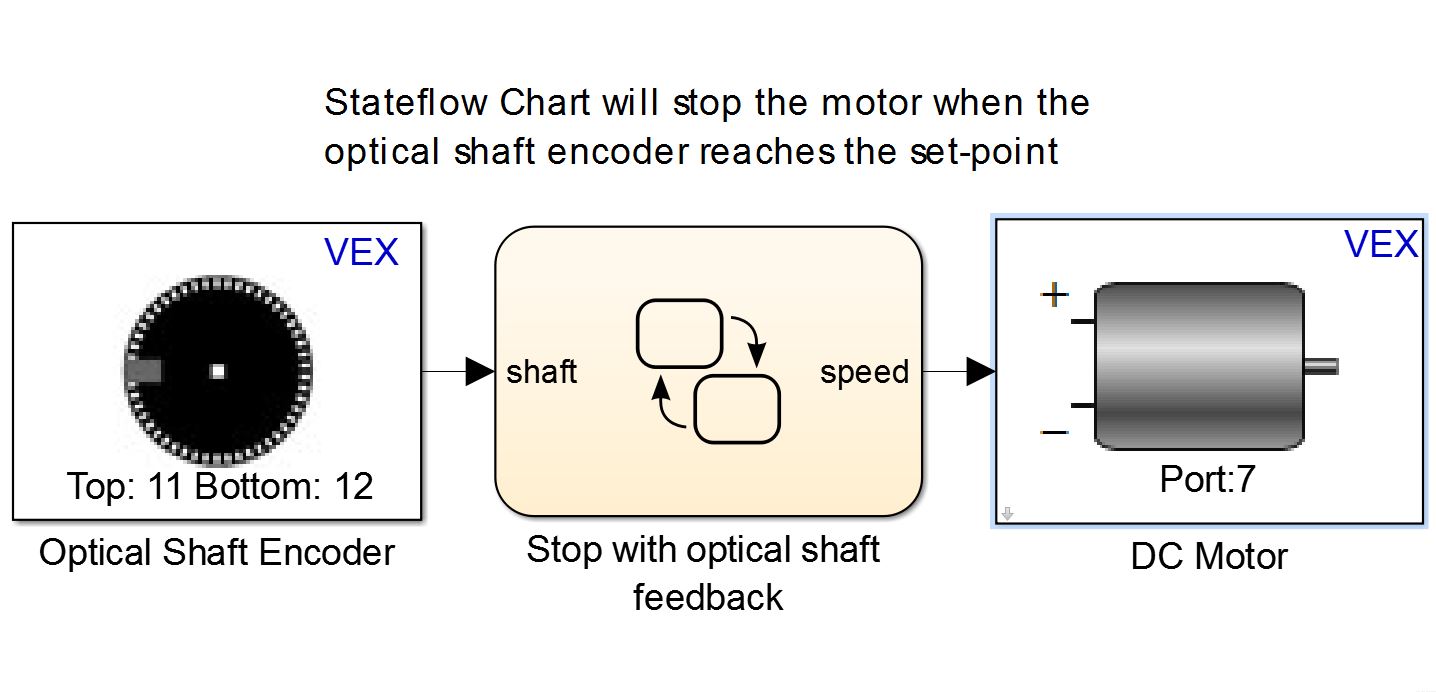
Optical Shaft Encoder in VEX Companion Library

Make sure the Block parameters are setup as in the following image. Remember you will have to set both the input and output ports in your block parameters to where you have physically plugged the optical shaft encoder into your VEX Cortex.



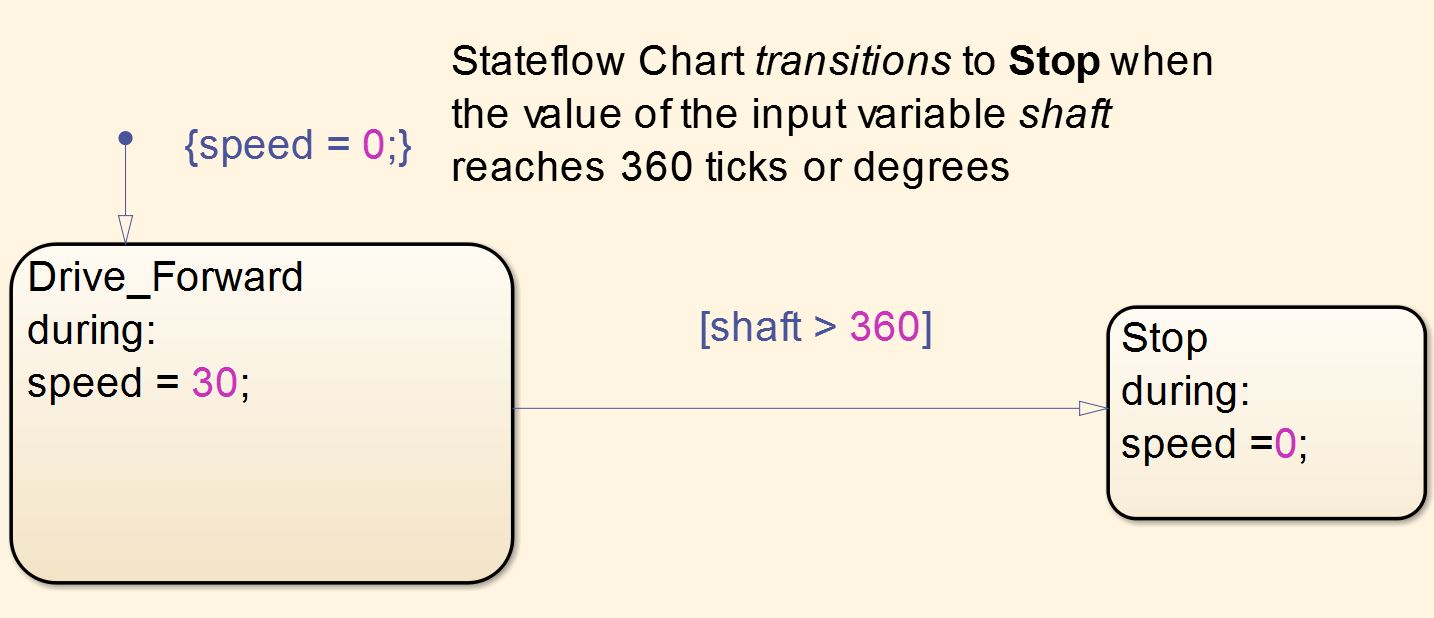
Optical Shaft Encoder Parameters

One major difference with the Optical Shaft Encoder is that the output actually measures degrees or rotation as opposed to ticks. Additionally, how you mount the Optical Shaft Encoder on the shaft is critical. If you mount it backwards, then your Optical Shaft Encoder will be measuring in the wrong direction.

We will also create a Stateflow Chart and the motor blocks onto our Simulink diagram. The Stateflow Chart will shut the motor off when it reaches a certain Tick value. The output of the Stateflow Chart goes to the Motor BlocSimulink Diagram using the Optical Shaft Encoder

We can also add additional blocks so we can run a simulation of our model before we actually deploy it to the hardware as you can see in the below image. You can refer to this model in the file called OpticalShaftModel.slx.

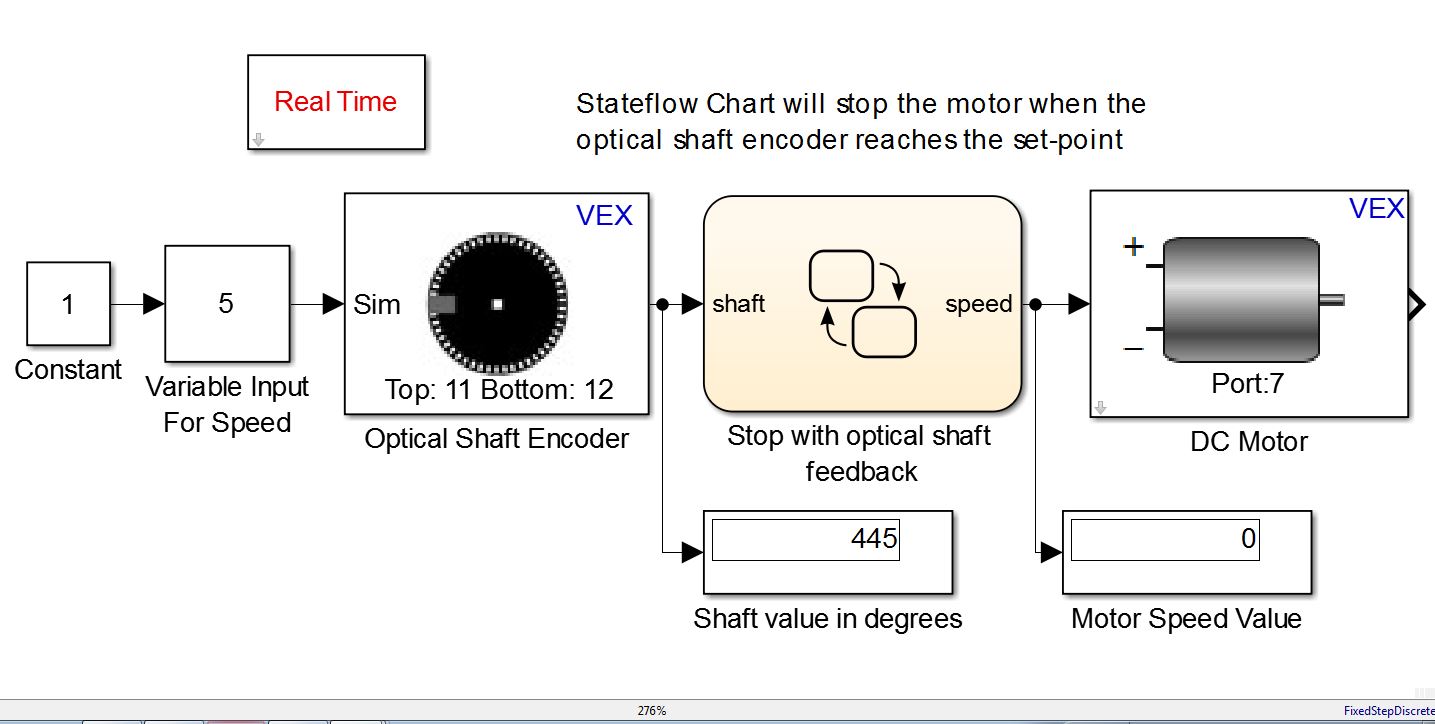
Looking inside of the Stateflow Chart we can see the simple commands. We have created a new input port called shaft.



Stateflow Chart using the Number of degrees of rotation as the Condition

When you deploy this program to your VEX Robot, you will find it will rotate until the shaft encoder reaches the desired value.

We can also add extra blocks to our Simulink diagram so we can run a simulation of the model before we deploy it to our robot. You can refer to this model in the file called OpticalShaftModelSimulation.slx.



Simulink diagram to model a simulation of our robot

In this section we have learned how to use the Optical Shaft Encoder.

Additional challenge: Similar to the last section, create a model where the robot drives in a square, but instead of using time as the condition in your Stateflow Chart, use the degrees of rotation value of the Optical Shaft Encoder.