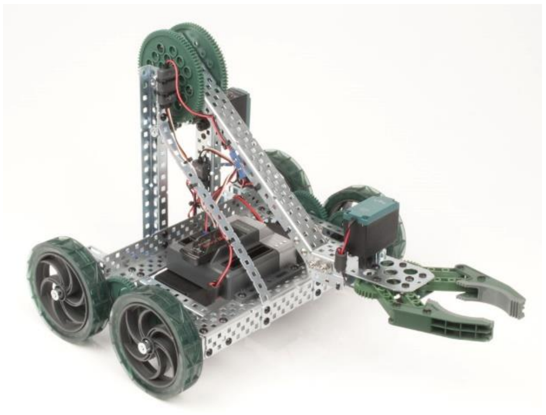
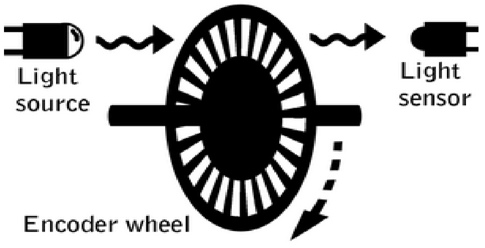
Marist School Robotics Teams
Directions for Writing Encoder Functions
ClawBot Training Robot

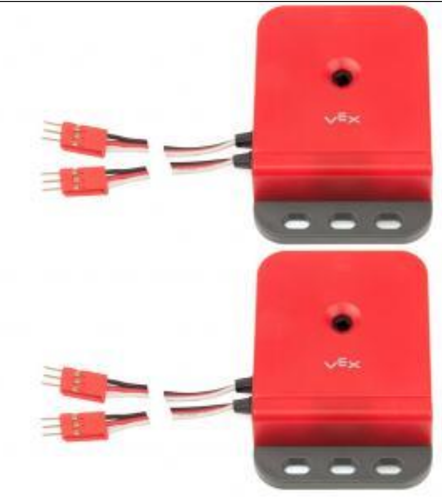


Description:

These directions will work through setting up the Pragma code and writing functions that use the Encoder sensors to measure the number of rotations of a motor shaft. An encoder has a black and white disk inside an enclosure with a light sensor. The light sensor ‘counts’ how many times the disk has spun and returns an integer to the program. These counts are usually called ‘ticks’ in a RobotC program.



There are two kinds of encoders in RobotC. The external Quadrature Encoder has a red casing and mounts to a shaft in the motor system. The external encoder uses two digital connections and plugs into two digital ports. The internal Encoder places the disk and light sensors directly into a VEX 393 motor and uses the I2C port to communicate with the VEX cortex.

 External Encoders (2 shown) Plug into digital ports (2 needed per encoder) Preferred use in VEX Robotics Competition	Internal Encoder attached to 393 Motor Uses 4 wire I2C connector to Vex Cortex

The forwardWithEncoder() function we write in this lesson will work with both types of encoders. The #Pragma setup for the external and internal encoders are different and both will be demonstrated.

Algorithm:

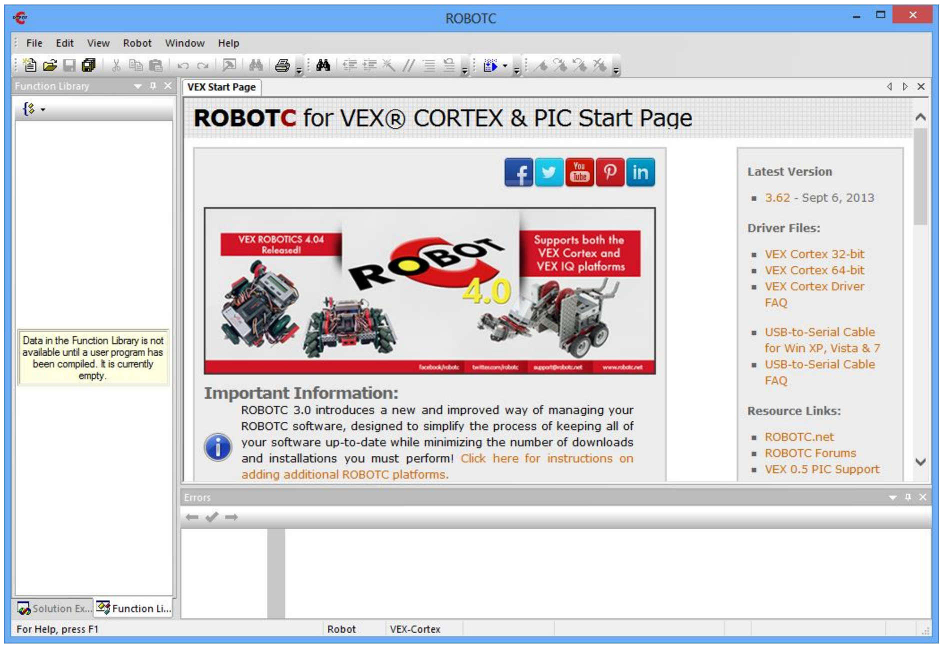
Like the touch sensor, the encoder follows the sensor based robotics algorithm. The algorithm for most sensor based robotics is:

0. Set up any variables or initialize sensors (Reset Encoders)
1. Start an action. (Turn on motors . . .)
2. Wait for a condition to be true. Measure the Encoder values. Make adjustments as needed.
3. When condition is met – stop the action from step 1.

You will note that the encoders will use the step 0.

Process:

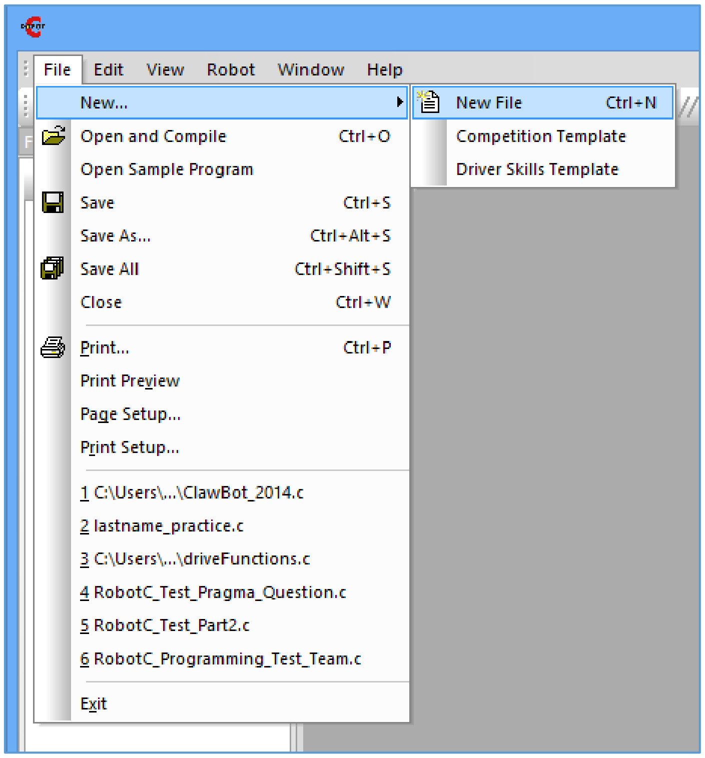
1. Start RobotC on your computer. (You may download RobotC at http://downloads.robotc.net/vex/ROBOTCforCortexPIC_362.exe)



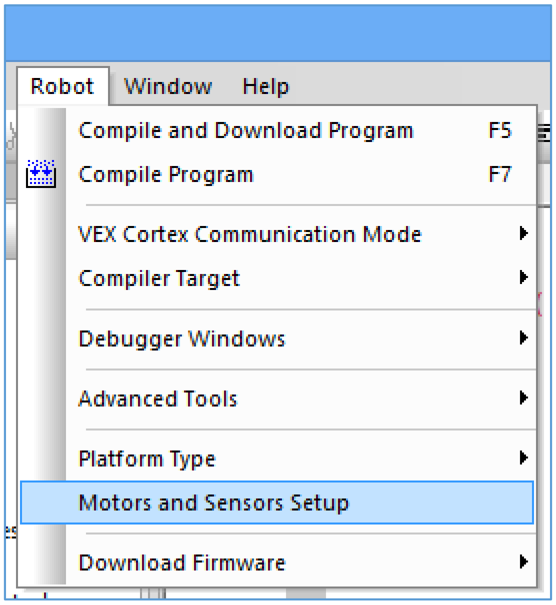
Data in the Function Library is not available until a user program has been compiled. It is currently empty.

Important Information:
ROBOTC 3.0 introduces a new and improved way of managing your ROBOTC software, designed to simplify the process of keeping all of your software up-to-date while minimizing the number of downloads and installations you must perform! Click here for instructions on adding additional ROBOTC platforms.

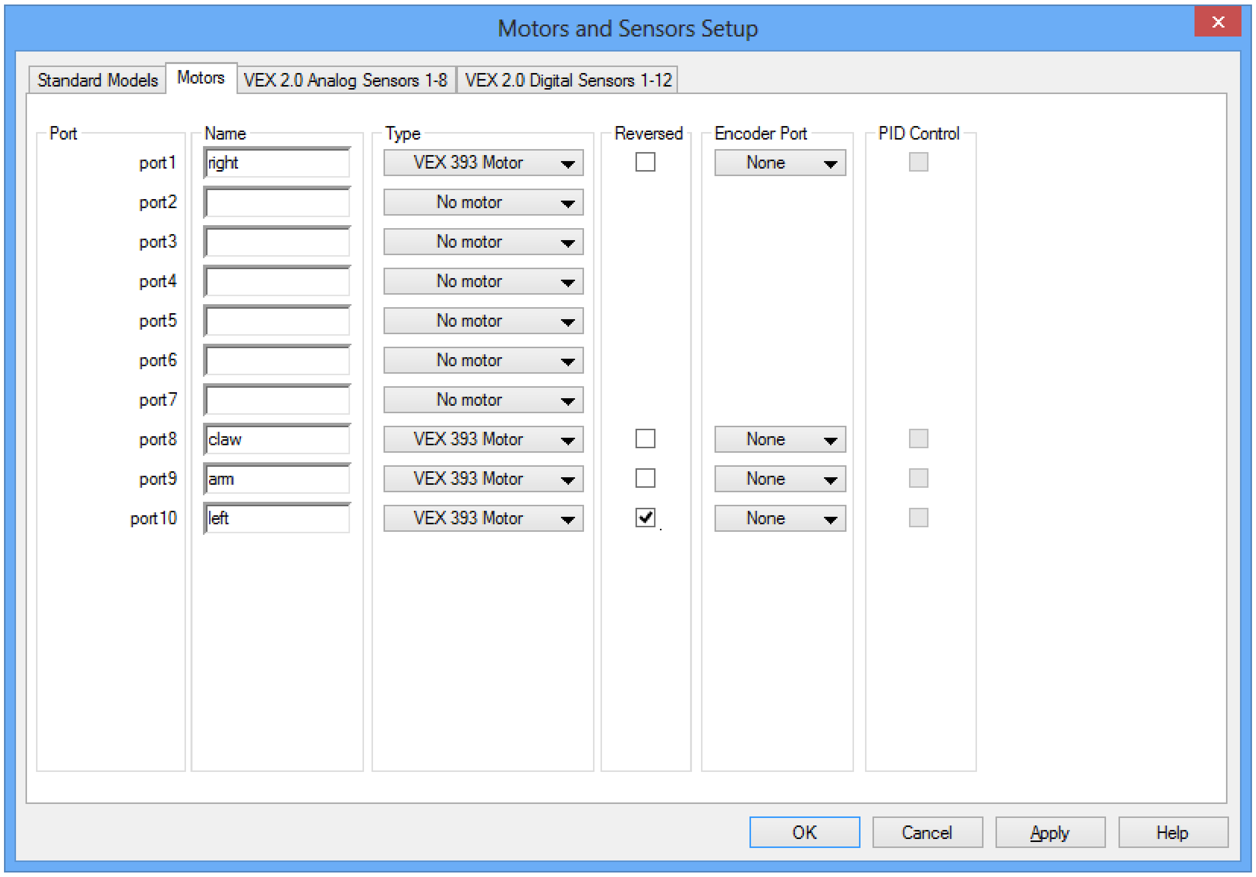
2. Select "File – New - NewFile" from the Menu Bar. This creates a new RobotC program.



3. We now need to set up the Motors and Sensors for the Robot. We will use the “Motors and Sensors” setup to set the Pragma Code for the program. Select “Robot – Motors and Sensors Setup” from the Menubar.



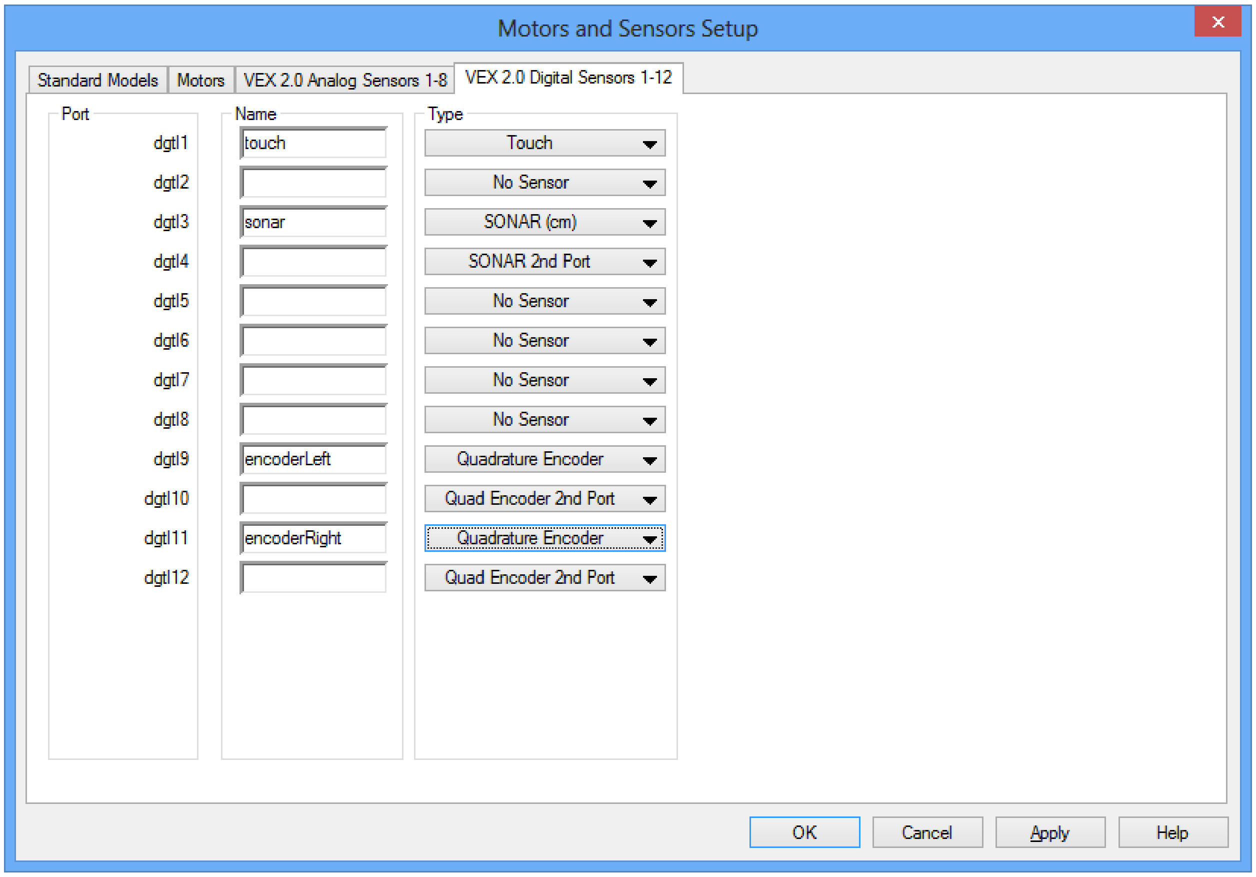
4. Click on the “Motors” tab and label the motors for the following:
   - -port1: right: 393
   - -port8: claw: 393
   - -port9: arm: 393
   - -port 10: left: 393: reversed



Port	Name	Type	Reversed	Encoder Port	PID Control
port1	right	VEX 393 Motor	☐	None	☐
port2		No motor			
port3		No motor			
port4		No motor			
port5		No motor			
port6		No motor			
port7		No motor			
port8	claw	VEX 393 Motor	☐	None	☐
port9	arm	VEX 393 Motor	☐	None	☐
port10	left	VEX 393 Motor	✓	None	☐

5a. (If you have External Encoders, follow this step) Click on the VEX 2.0 Digital Sensors 1-12 tab and set up the sensors:

-dgtl1: touch: Touch
-dgtl3: sonar: SONAR (cm)
-dgtl9: encoderLeft: Quadrature Encoder
-dgtl11: encoderRight: Quadrature Encoder

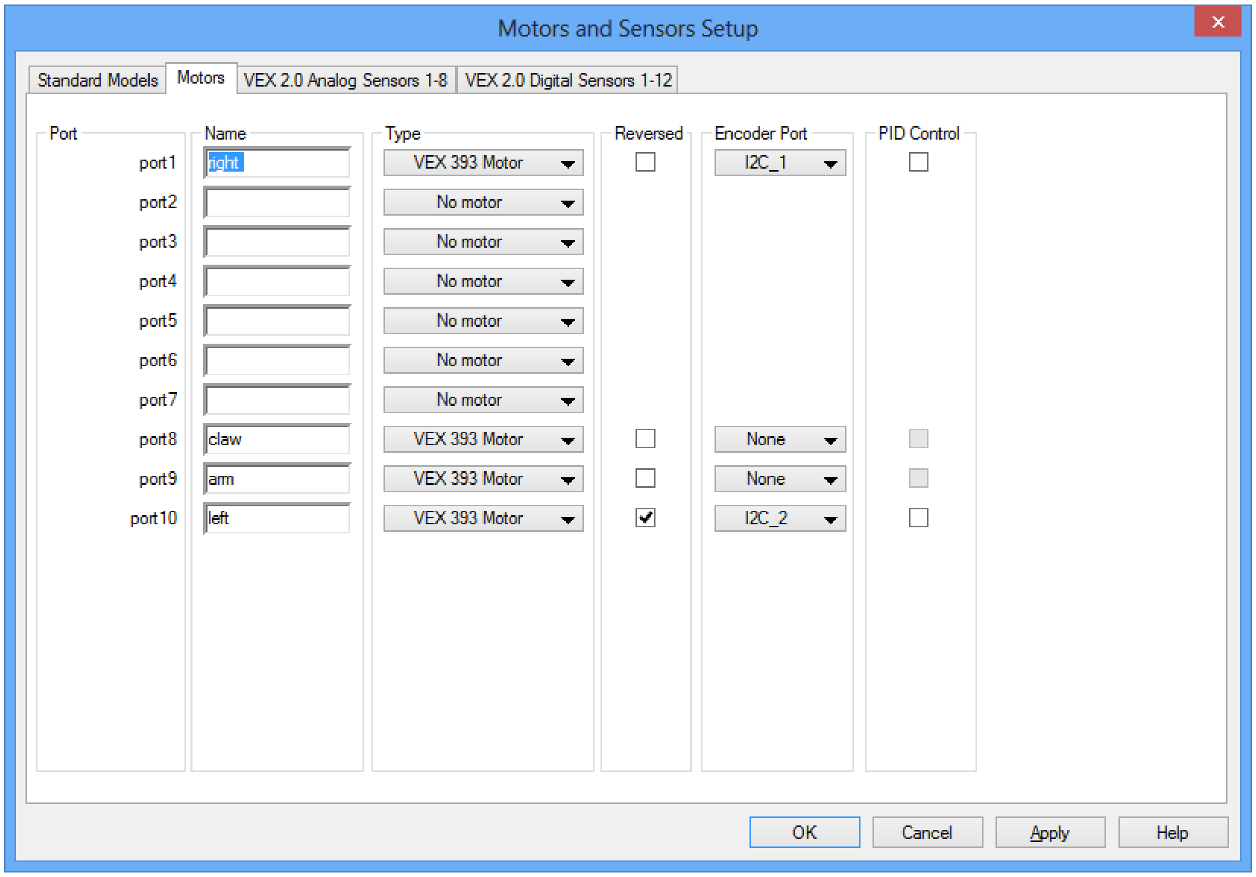


Port	Name	Type
dgtl1	touch	Touch
dgtl2		No Sensor
dgtl3	sonar	SONAR (cm)
dgtl4		SONAR 2nd Port
dgtl5		No Sensor
dgtl6		No Sensor
dgtl7		No Sensor
dgtl8		No Sensor
dgtl9	encoderLeft	Quadrature Encoder
dgtl10		Quad Encoder 2nd Port
dgtl11	encoderRight	Quadrature Encoder
dgtl12		Quad Encoder 2nd Port

5b. (If you have internal encoders, follow these steps)

a. Click on the VEX 2.0 Digital Sensors 1-12 tab and set up the sensors:
  -dgtl1: touch: Touch
  -dgtl3: sonar: SONAR (cm)

b. Click on the Motors tab and set Encoders for ports 1 and 10
  -port1 will be I2C_1
  -port2 will be I2C_2


Motors and Sensors Setup

Standard Models | Motors | VEX 2.0 Analog Sensors 1-8 | VEX 2.0 Digital Sensors 1-12

Port	Name	Type	Reversed	Encoder Port	PID Control
port1	right	VEX 393 Motor	☐	I2C_1	☐
port2		No motor			
port3		No motor			
port4		No motor			
port5		No motor			
port6		No motor			
port7		No motor			
port8	claw	VEX 393 Motor	☐	None	☐
port9	arm	VEX 393 Motor	☐	None	☐
port10	left	VEX 393 Motor	✓	I2C_2	☐

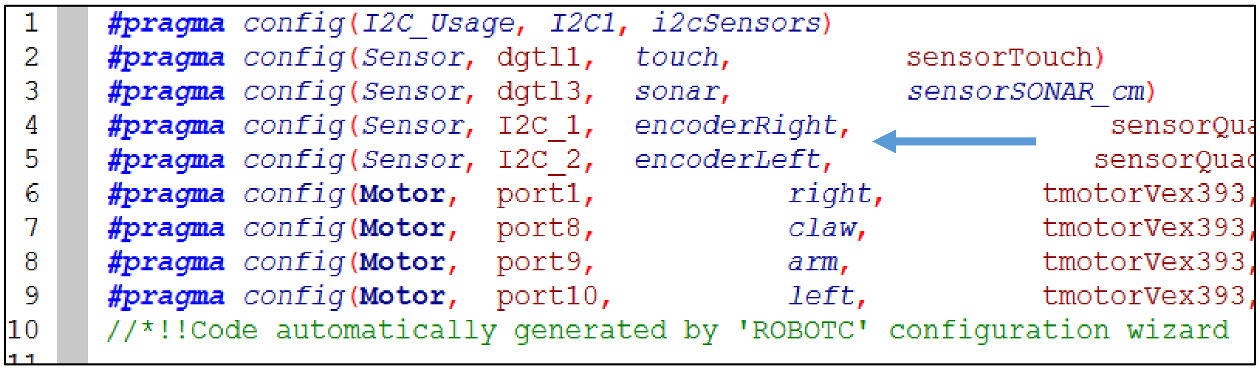
OK | Cancel | Apply | Help

c. Click 'Apply' and 'OK'

d. Label the I2C ports in the pragma with:

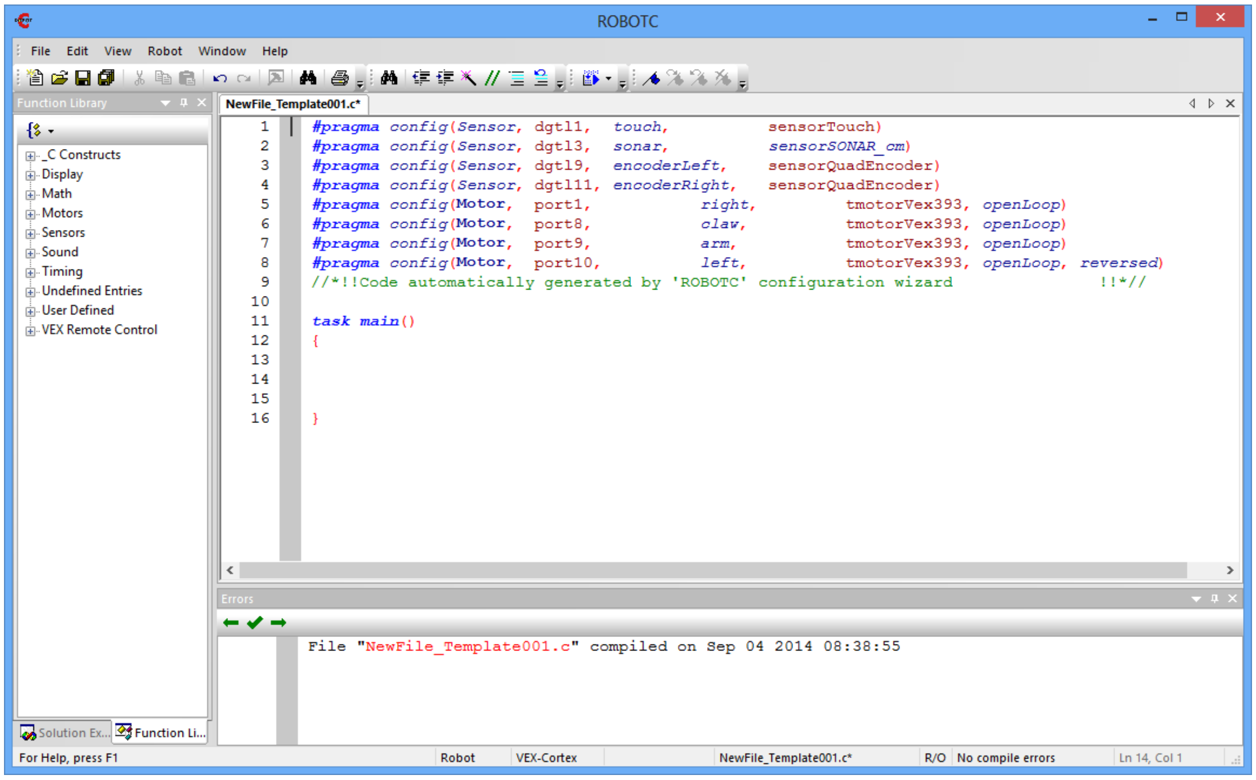
-I2C_1: encoderRight

-I2C_2: encoderLeft



```
#pragma config(I2C_Usage, I2C1, i2cSensors)
#pragma config(Sensor, dgtl1,   touch,          sensorTouch)
#pragma config(Sensor, dgtl3,   sonar,          sensorSONAR_cm)
#pragma config(Sensor, I2C_1,   encoderRight,               sensorQu
#pragma config(Sensor, I2C_2,   encoderLeft,               sensorQuad
#pragma config(Motor,  port1,           right,         tmotorVex393,
#pragma config(Motor,  port8,           claw,          tmotorVex393,
#pragma config(Motor,  port9,           arm,           tmotorVex393,
#pragma config(Motor,  port10,          left,          tmotorVex393,
//*!!Code automatically generated by 'ROBOTC' configuration wizard
```

6. Your code should look like this:

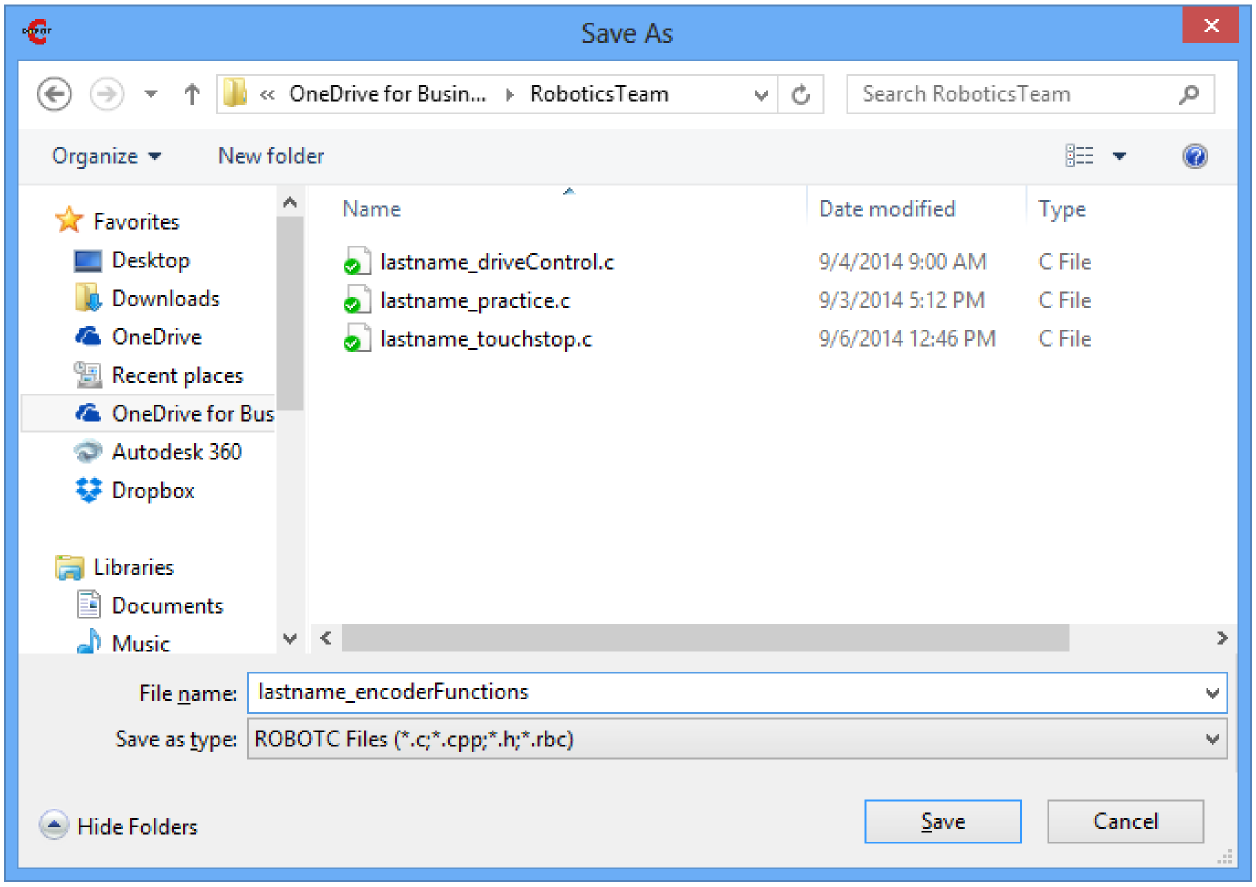


```
#pragma config(Sensor, dgtl1,   touch,          sensorTouch)
#pragma config(Sensor, dgtl3,   sonar,          sensorSONAR_cm)
#pragma config(Sensor, dgtl9,   encoderLeft,    sensorQuadEncoder)
#pragma config(Sensor, dgtl11,  encoderRight,   sensorQuadEncoder)
#pragma config(Motor,  port1,           right,         tmotorVex393, openLoop)
#pragma config(Motor,  port8,           claw,          tmotorVex393, openLoop)
#pragma config(Motor,  port9,           arm,           tmotorVex393, openLoop)
#pragma config(Motor,  port10,          left,          tmotorVex393, openLoop, reversed)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

task main()
{



}
```

```
File "NewFile_Template001.c" compiled on Sep 04 2014 08:38:55
```

For Help, press F1 | Robot | VEX-Cortex | NewFile_Template001.c* | R/O | No compile errors | Ln 14, Col 1

7. Select “File – Save” and save your program to your OneDrive for Business as “lastname_encoderFunctions”

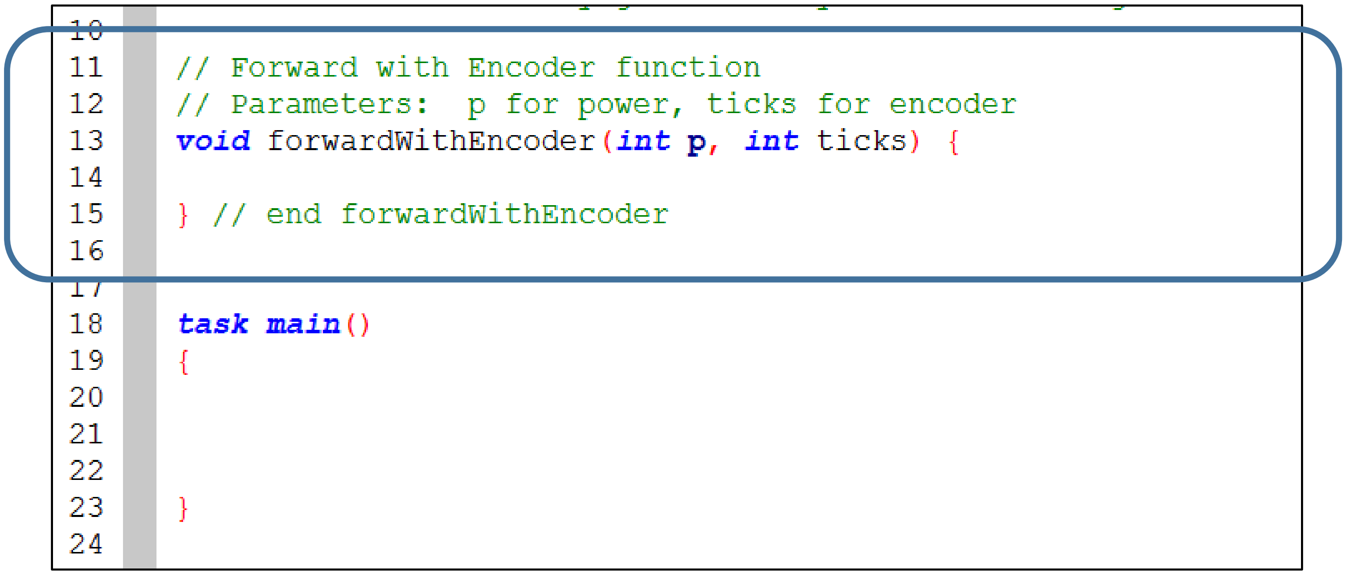


Name	Date modified	Type
lastname_driveControl.c	9/4/2014 9:00 AM	C File
lastname_practice.c	9/3/2014 5:12 PM	C File
lastname_touchstop.c	9/6/2014 12:46 PM	C File

File name: lastname_encoderFunctions

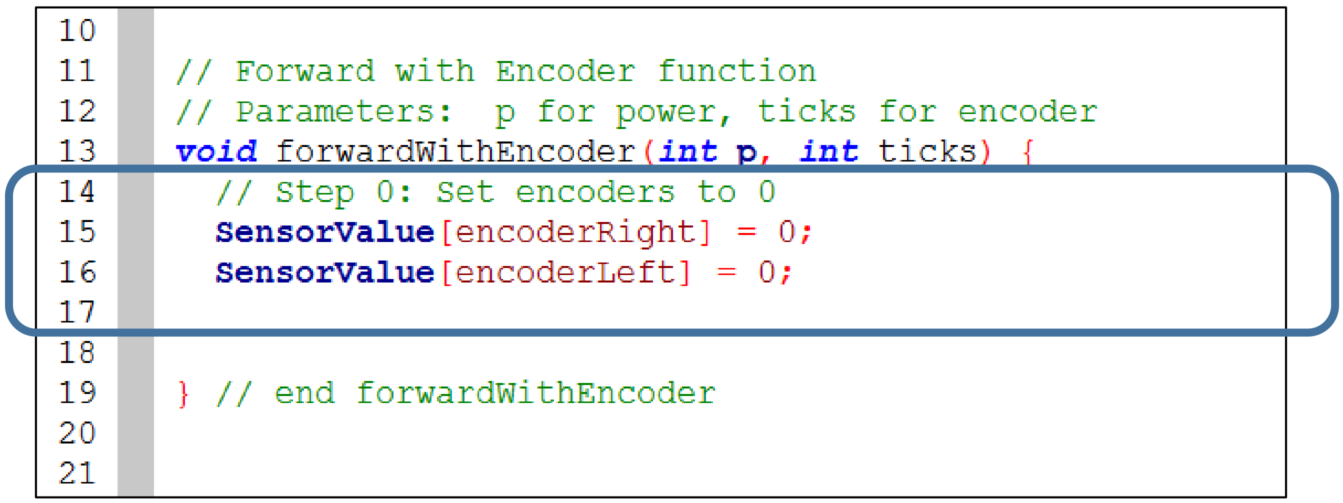
Save as type: ROBOTC Files (*.c;*.cpp;*.h;*.rbc)

8. Begin to define the structure for the forwardWithEncoderFunction()



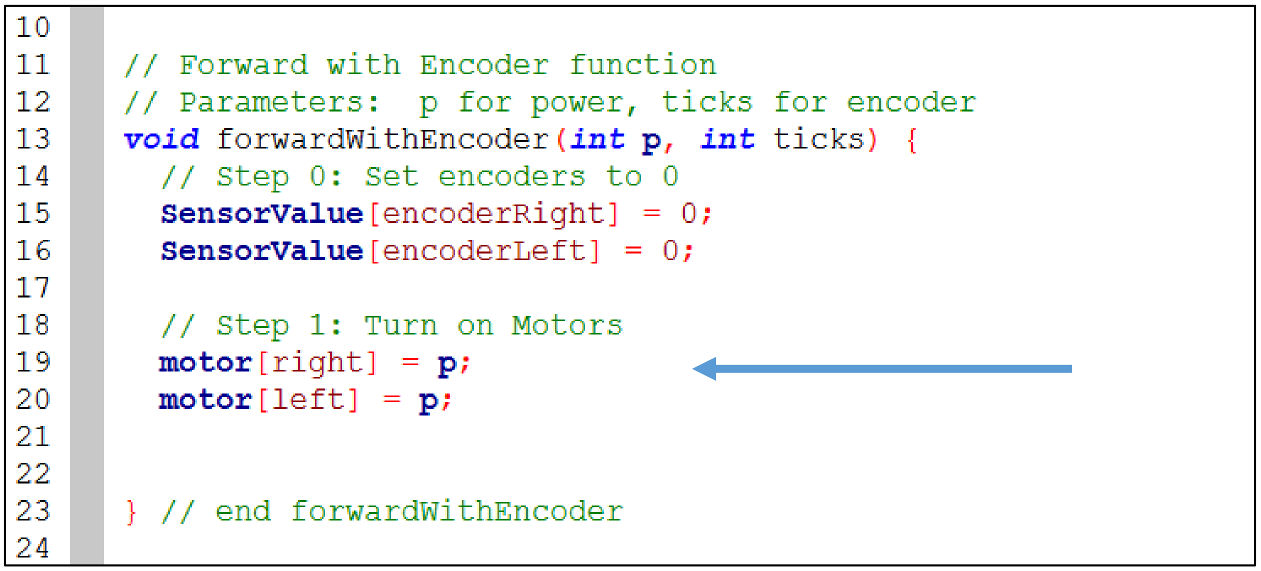
```
// Forward with Encoder function
// Parameters:  p for power, ticks for encoder
void forwardWithEncoder(int p, int ticks) {

} // end forwardWithEncoder

task main()
{



}
```

9. Place the code for step 1: Initialize the encoders in the function:



```

// Forward with Encoder function
// Parameters:  p for power, ticks for encoder
void forwardWithEncoder(int p, int ticks) {
  // Step 0: Set encoders to 0
  SensorValue[encoderRight] = 0;
  SensorValue[encoderLeft] = 0;


} // end forwardWithEncoder

```

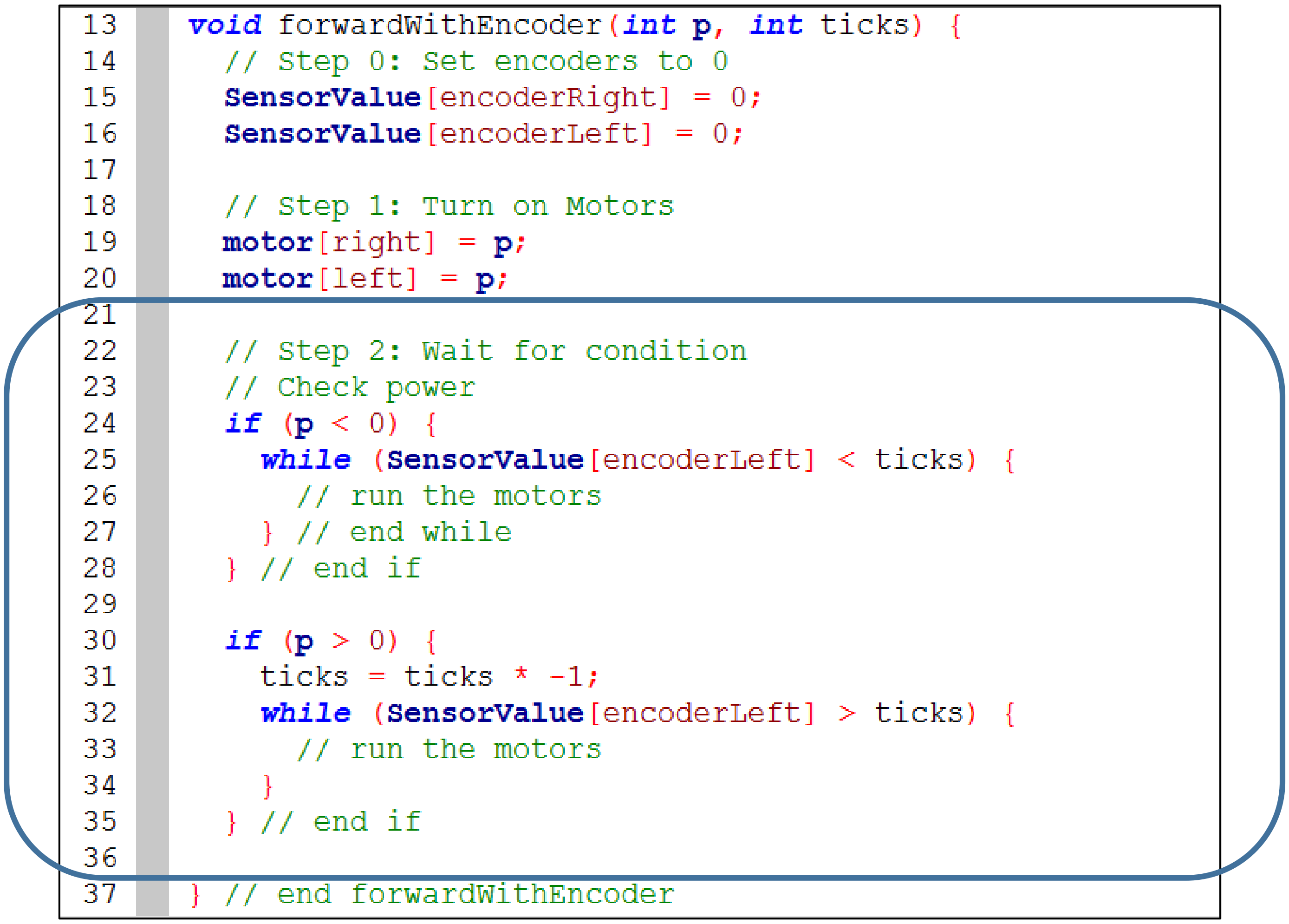
10. For step 1, set the power of the motors to parameter p.



```

// Forward with Encoder function
// Parameters:  p for power, ticks for encoder
void forwardWithEncoder(int p, int ticks) {
  // Step 0: Set encoders to 0
  SensorValue[encoderRight] = 0;
  SensorValue[encoderLeft] = 0;

  // Step 1: Turn on Motors
  motor[right] = p;
  motor[left] = p;


} // end forwardWithEncoder

```

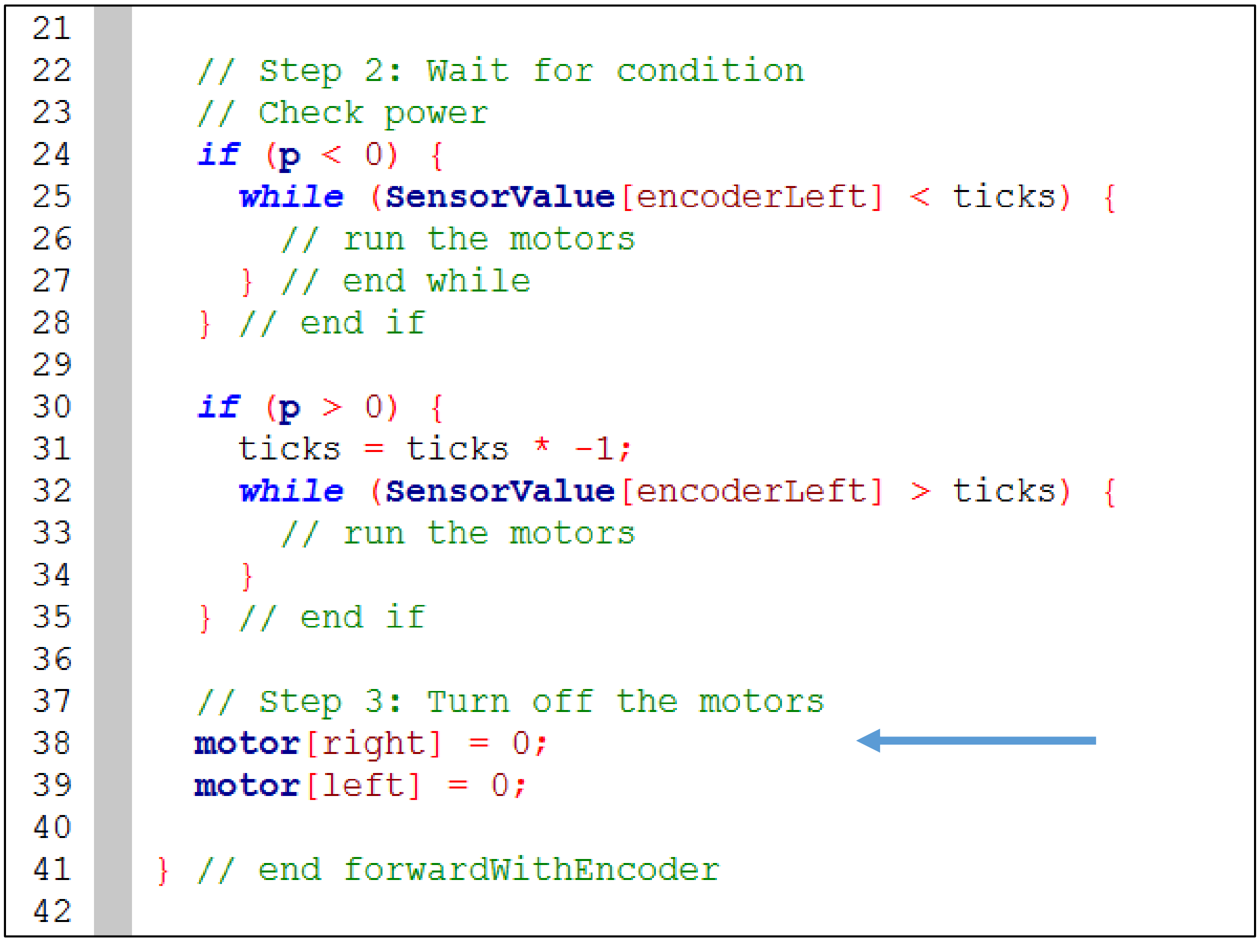
11. Step 2, we need to account for if the power is greater or less than 0 (Robot is moving backwards or forwards). The encoders account for positive and negative motion (with a range of about -32,000 to 32,000). We will use if statements and two whiles to accomplish this.

Keep in mind that when you test this code, if the function does not work, you might need to experiment with the if statements and operators.



```
void forwardWithEncoder(int p, int ticks) {
  // Step 0: Set encoders to 0
  SensorValue[encoderRight] = 0;
  SensorValue[encoderLeft] = 0;

  // Step 1: Turn on Motors
  motor[right] = p;
  motor[left] = p;

  // Step 2: Wait for condition
  // Check power
  if (p < 0) {
    while (SensorValue[encoderLeft] < ticks) {
      // run the motors
    } // end while
  } // end if

  if (p > 0) {
    ticks = ticks * -1;
    while (SensorValue[encoderLeft] > ticks) {
      // run the motors
    }
  } // end if

} // end forwardWithEncoder
```

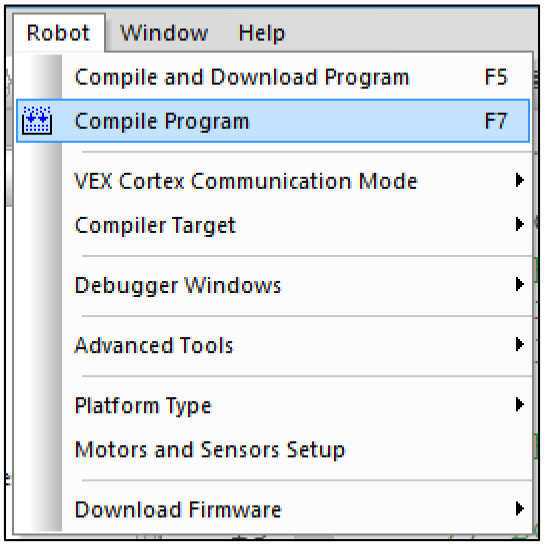
12. We will end the function with Step 3: turn off the motors.



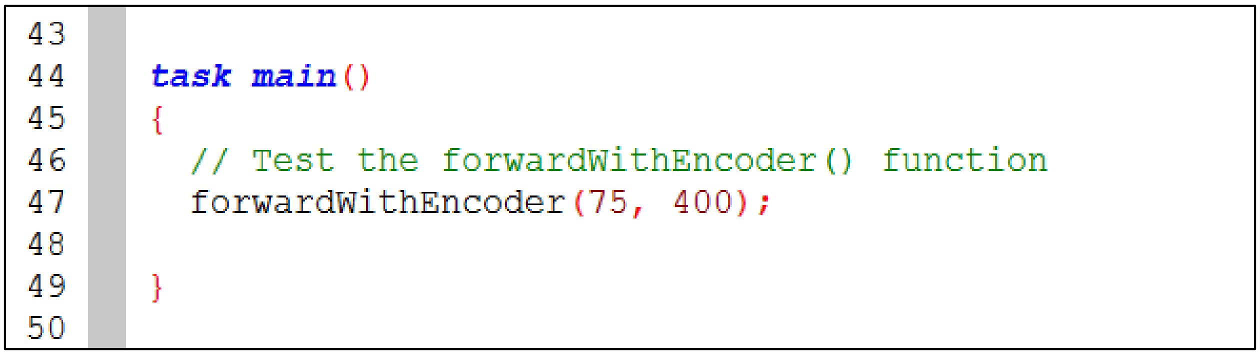
```

  // Step 2: Wait for condition
  // Check power
  if (p < 0) {
    while (SensorValue[encoderLeft] < ticks) {
      // run the motors
    } // end while
  } // end if

  if (p > 0) {
    ticks = ticks * -1;
    while (SensorValue[encoderLeft] > ticks) {
      // run the motors
    }
  } // end if

  // Step 3: Turn off the motors
  motor[right] = 0;
  motor[left] = 0;

} // end forwardWithEncoder

```

13. Save your work by selecting ‘File – Save’.

14. Select ‘Robot – Compile Program’ to compile and check for errors.

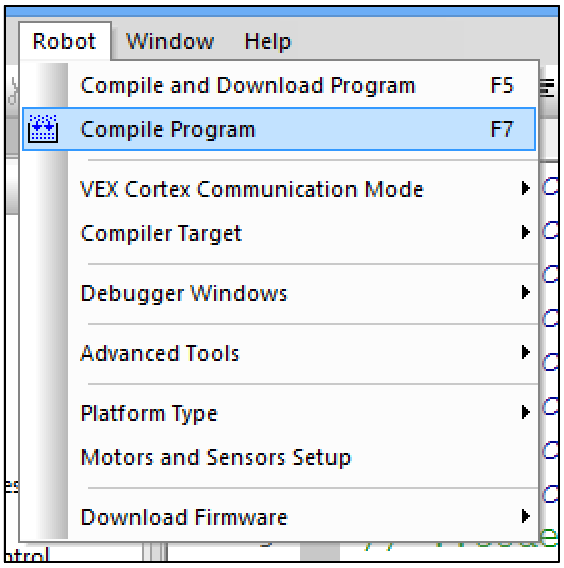


15. We need to test the forwardWithEncoder() function. In the task main() call the forwardWithEncoder() function with a power level of 75 and a ticks value of 400.



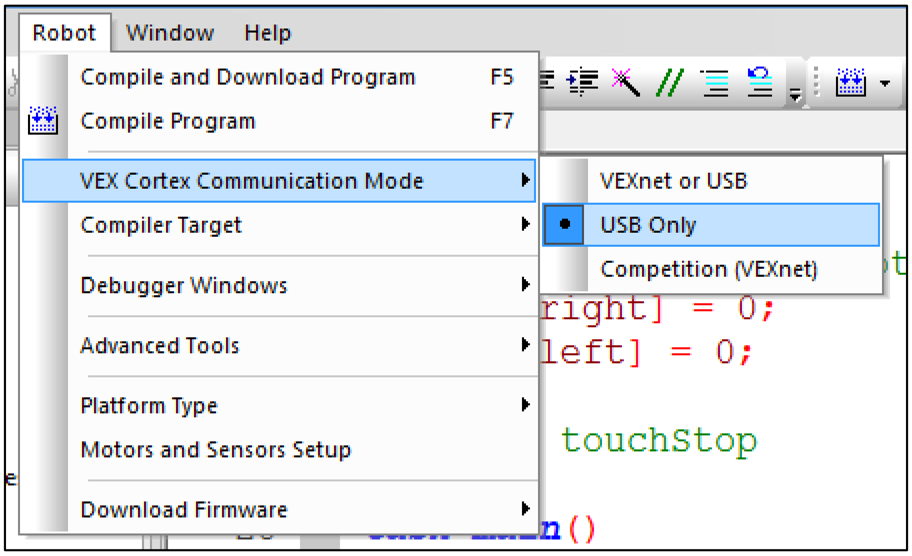
```

task main()
{
  // Test the forwardWithEncoder() function
  forwardWithEncoder(75, 400);

}

```

16. To check for errors, select “Robot-Compile Program” from the Menu Bar.



17. If the program compiles – you will not have errors (red x marks next the numbers). Now it is time to load the program in the Robot.

18. First, set the communication mode for the Robot to be “USB Only” by going to ‘Robot – VEX Cortex Communication Mode – USB Only’ on the Menu bar. This allows the robot to run the program without having to plug into the computer or a Vex Remote.

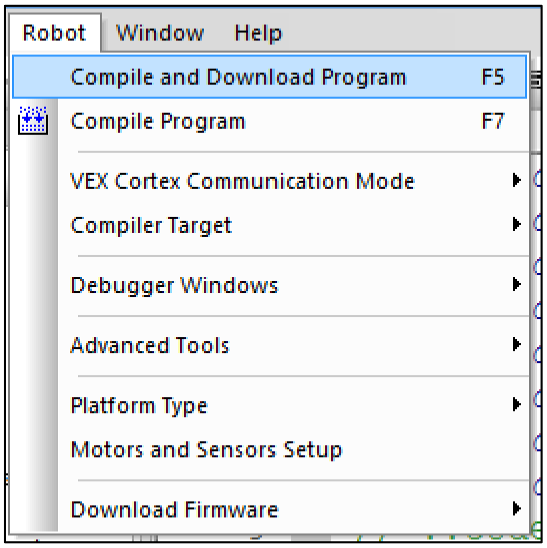


19. Take your ClawBot and make sure you have the following:
    - -ClawBot
    - -Orange USB Cable
    - -Blue Battery Pack (7.2 Volts, 3000 mAH)
    - -Confirm that the external encoders are plugged into ports 9, 10, 11, and 12

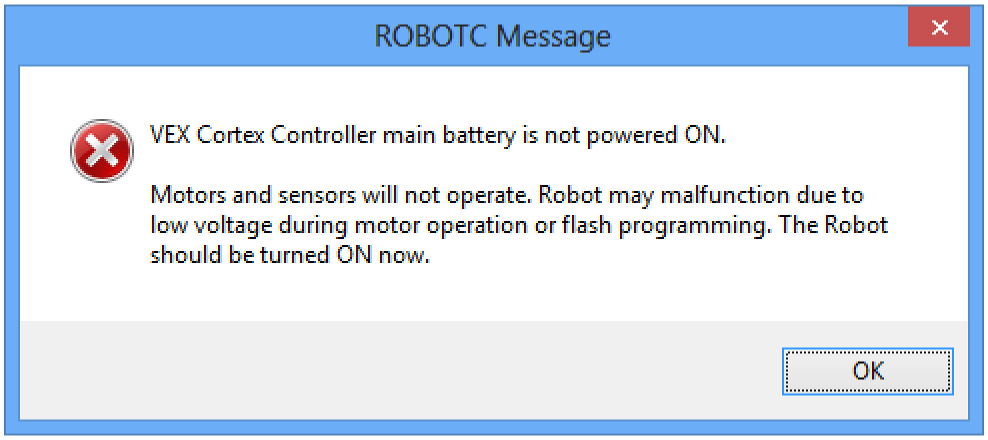
20. Battery into the VEX Cortex brick but do not turn on.

21. Plug the orange USB cable into the Cortex and into your computer.

22. Select "Robot – Compile and Download Program" from the Menu Bar.



23. You might get a message about the Robot not being switched on. Click OK at this message:

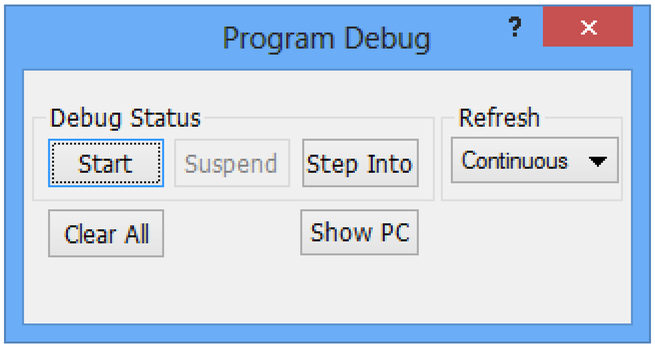


ROBOTC Message

VEX Cortex Controller main battery is not powered ON.

Motors and sensors will not operate. Robot may malfunction due to low voltage during motor operation or flash programming. The Robot should be turned ON now.

24. The Program Debug window will open.   To test the robot at this stage, hold the robot in the air and be careful not to have wires in fingers in the gears. (Propping the Robot on a stack of books works well for ‘bench’ testing).   Turn the robot on. Press the ‘Start’ button icon on the ‘Program Debug’ window to start the program. Until the encoder reaches 400. (About 1 rotation).



25. To test the robot without the orange cable; Close the ‘Program Debug’ window. Turn the robot off. Unplug the robot from the computer. Place the robot in an open space and turn the robot on. It should run the program directly.

26. If the robot moves forward and stops after a certain distance – you have passed. Try different values for ticks in the task main() to experiment.