About ROBOTC
ROBOTC
Programming software for the VEX® CORTEX and VEX® IQ Robotics Systems
ROBOT
Program Smart.
Robomatter INCORPORATED
Manufactured with Intellectual properties from Carnegie Mellon University

Introduction to Programming

VEX IQ with ROBOTC4

Users Guide

Table of Contents

Frequently Asked Questions (FAQ)

Before starting

- ► What do I need to do before I begin programming my VEX IQ?
  See "Checklist", page 4.
- ► What is ROBOTC Graphical Language?
  See "What is ROBOTC Graphical Language", page 5.
- ► How should my robot be configured?
  See "Proper Motor Setup for Standard Build", page 6.
  See "Initial Wireless Setup", page 9.
- ► What is the difference between Firmware and Software?
  See "VEX IQ Firmware", page 7.
  See "Install ROBOTC Graphical Software on the VEX IQ", page 8
- ► How do I Upload Code to my robot?
  See "Uploading Autonomous verse Remote Control Programs", page 10
  See "Troubleshooting", page 11.

Programming

- ► How do I start a new program?
  See "How do I start my first program", page 12.
- ► How do I write my first program?
  See "How do I write a program", page 13
- ► How do I save my programs?
  See "How do I save/open a program", page 14
- ► I save my program into Auto Programs and it saves into TeleOp Programs
  See page 10, Uploading Autonomous verse Remote Control programs
- ► What if my program contains errors?
  See "How do I edit a program", see page 15.
- ► What are the commands of ROBOTC Graphical Language?
  See "ROBOTC Graphical Language Commands", page 16.
- ► Can I see what my code looks like in full ROBOTC?
  See "Toggling between Graphical and Text-based code", see page 5.

Try Me Example Code

- ► ROBOTC Graphical Code Examples

ROBOTC's Graphical Programming Language is designed to be intuitive. Starting on page 16 are examples of working code that we encourage new users to test so they can become familiar with the simplicity of this software.

Checklist

- ☐ **Charge your battery and your VEX IQ Remote Control**
  Once you get your robot built you will want to begin playing with it.

- ☐ **Build your robot correctly**
  ROBOTC's Graphical Language expects to see the standard build with the left motor connected to port1 and the right motor connected to port6. It is possible to program using ROBOTC Graphical Language with other motor configurations, but beginners should start with the standard build. See page 6.

- ☐ **Update your firmware**
  See page 7.

- ☐ **Download ROBOTC4 Graphical Language**
  Make sure that you have the latest build, www.robotc.net.

- ☐ **Check your initial wireless setup**. See page 9.

- ☐ **Test some simple code**
  See page 10 uploading Autonomous verse uploading Remote Control Programs

- ☐ **If you are using this with a classroom be prepared to show them how and where to save their programs.**
  This is important because kids will forget where they saved their work. Give them a set of directions (i.e. make a folder on your desktop). See page 14

- ☐ **Clear an area to work**
  Robots often misbehave and need a bit of extra room to test.

Get Started!

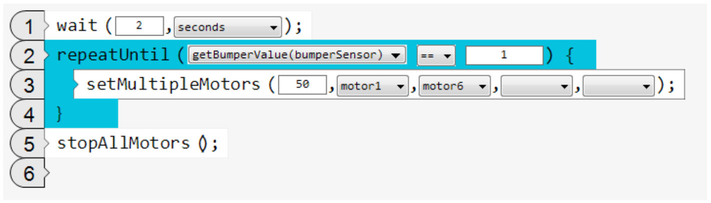
ROBOTC Graphical Language

What is the ROBOTC Graphical Language for VEX IQ and why should I use it?

ROBOTC Graphical Language is an easy to use drop and drag programming language for beginning programmers. Students don't have to worry about the locations of semicolons, curly braces, brackets, and other general syntax errors, but they are able to see where they should be when they transition to full ROBOTC.

ROBOTC Graphical → ***ROBOTC NL*** → ***ROBOTC***

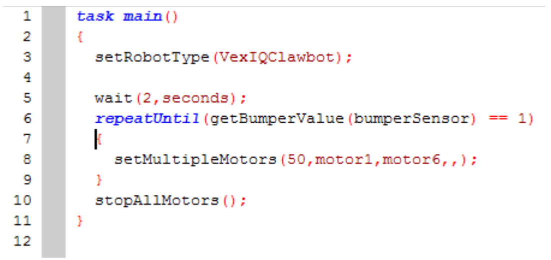
ROBOTC Graphical



```
wait ( 2 , seconds );
repeatUntil ( getBumperValue(bumperSensor) == 1 ) {
    setMultipleMotors ( 50 , motor1 , motor6 , , );
}
stopAllMotors ();
```

The graphical language is designed to allow students to quickly get started.

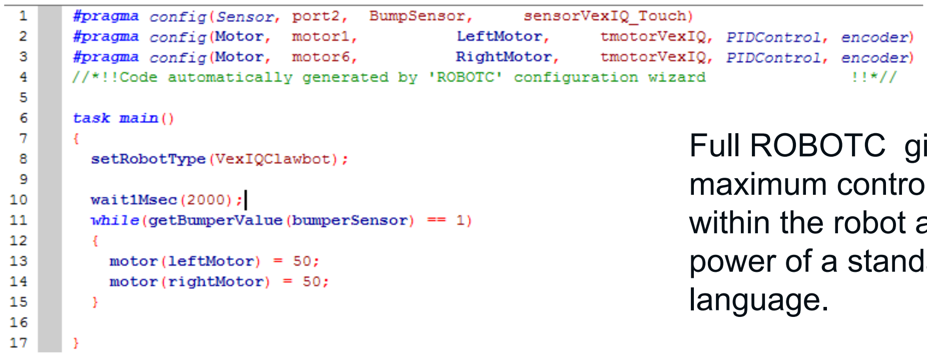
ROBOTC Natural Language



```
task main()
{
  setRobotType(VexIQClawbot);

  wait(2,seconds);
  repeatUntil(getBumperValue(bumperSensor) == 1)
  {
    setMultipleMotors(50,motor1,motor6,,);
  }
  stopAllMotors();
}
```

Natural Language gives students more control and uses pseudocode like commands but still required students to understand basic syntax rules.

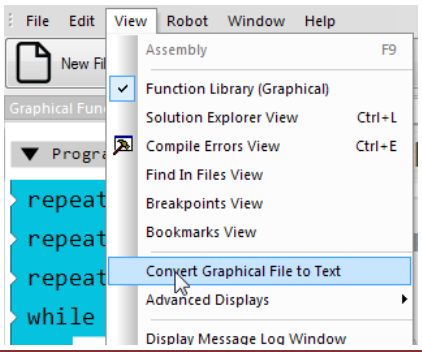
Full ROBOTC



```
#pragma config(Sensor, port2,  BumpSensor,     sensorVexIQ_Touch)
#pragma config(Motor,  motor1,          LeftMotor,     tmotorVexIQ, PIDControl, encoder)
#pragma config(Motor,  motor6,          RightMotor,    tmotorVexIQ, PIDControl, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

task main()
{
  setRobotType(VexIQClawbot);

  wait1Msec(2000);
  while(getBumperValue(bumperSensor) == 1)
  {
    motor(leftMotor) = 50;
    motor(rightMotor) = 50;
  }

}
```

Full ROBOTC gives the student maximum control of all systems within the robot as well as the full power of a standard programming language.

Convert Graphical to ROBOTC code



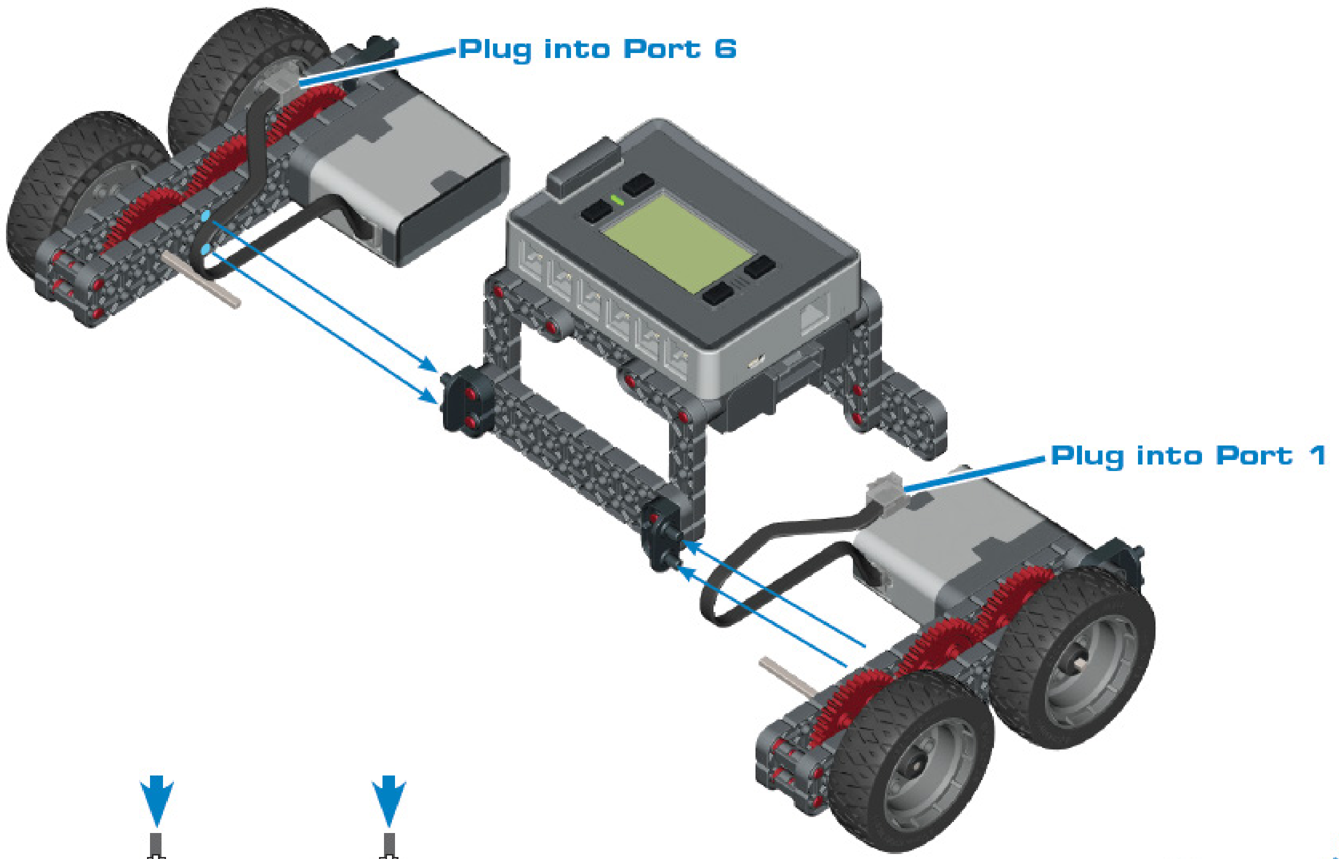
To see what your graphical code would look like in standard ROBOTC begin by saving you file so that you have a saved copy. Then:

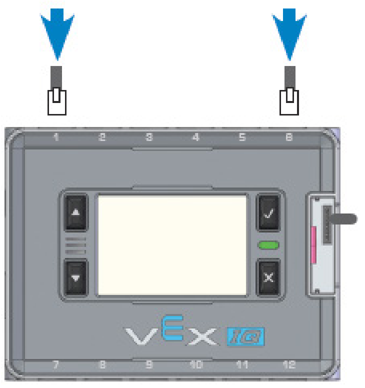
- Select View
- Select Convert Graphical File to Text

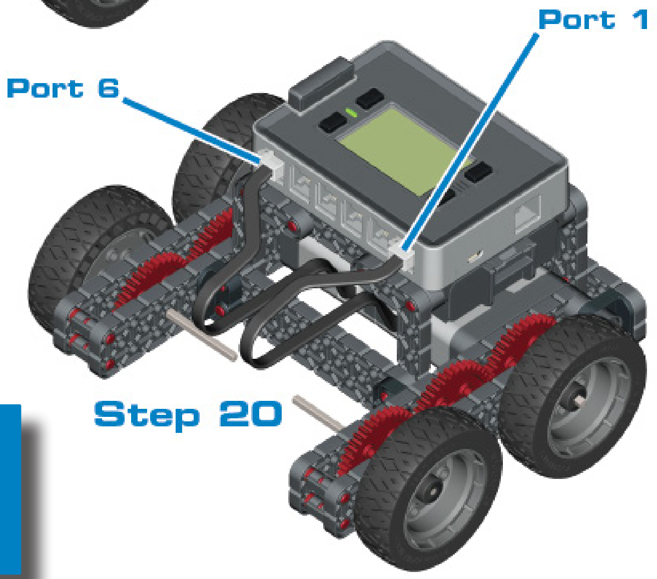
Note: you can go from graphical to text, but not from text to graphical.

Standard Build Proper Motor Setup

In order to insure maximum compatibility with ROBOTC's Graphical Language commands your robot's motors should be plugged into Ports 1 & 6 as shown below.



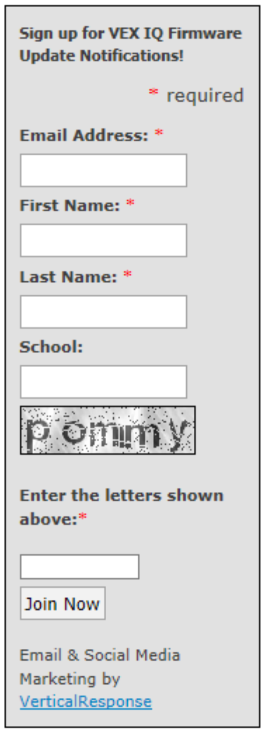




Type VEX IQ Building Instructions into a search engine to find a full set of building instructions for your VEX IQ robot.

VEX IQ Firmware

Note: All of this information on this page is taken directly from the Innovation First VEX IQ website. It is important to sign up at the VEX IQ website to insure that you are using the latest firmware



Sign up for VEX IQ Firmware Update Notifications!

* required

Email Address: *

First Name: *

Last Name: *

School:

pommy

Enter the letters shown above:*

Join Now

Email & Social Media Marketing by VerticalResponse

VEX IQ Firmware Updates

http://www.vexrobotics.com/vexiq/software/firmware

All of the VEX IQ Smart Devices (Robot Brain, Controller, Smart Motor, and sensors) contain their own internal processors and run special software called firmware. This firmware is what allows for advanced programming features and an enhanced user experience. The best way to ensure that your VEX IQ system is functioning properly is to keep the firmware up to date.

Important Note: When the firmware on the Robot Brain is updated, all sensors and motors must also be updated by plugging them in to the Brain during the update.

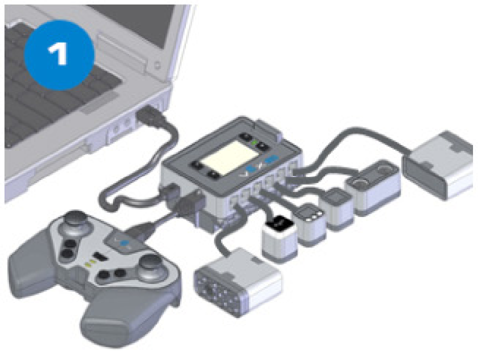
Installing the VEX IQ Firmware Updater

1. Click on the download button below to download the installer.

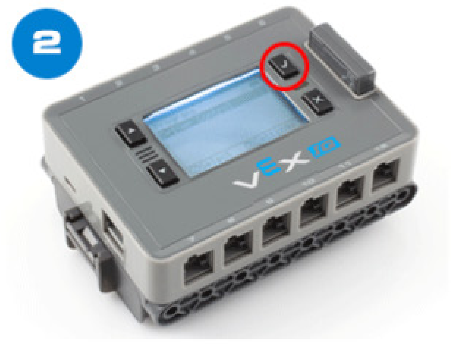
Note: Go to http://www.vexrobotics.com/vexiq/software/firmware to find the latest firmware.

2. If given the option, choose “Run”. Otherwise, save the file and open it. Follow the on-screen instructions to install the VEX IQ Firmware Updater preview.

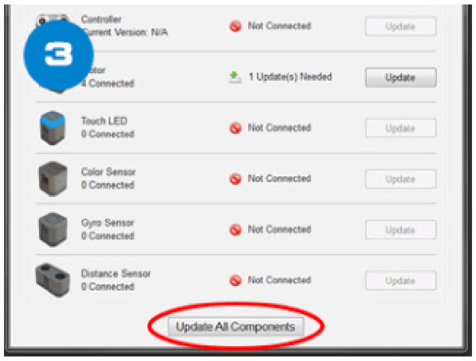
Using the VEX IQ Firmware Updater - 3 Easy Steps!



1 Plug all devices into the Robot Brain, and plug the Robot Brain into your computer via USB.



2 Turn on the Robot Brain.



Controller
Current Version: N/A — Not Connected — Update

Motor
4 Connected — 1 Update(s) Needed — Update

Touch LED
0 Connected — Not Connected — Update

Color Sensor
0 Connected — Not Connected — Update

Gyro Sensor
0 Connected — Not Connected — Update

Distance Sensor
0 Connected — Not Connected — Update

Update All Components

3 Open the VEX IQ Firmware Updater, and click “Update All Components”.

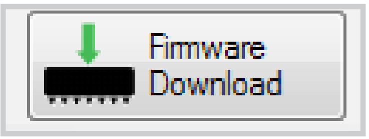
Install ROBOTC Graphical on VEX IQ

Items you will need:

- Robot Brain with Radio and Robot Battery installed
- VEX Controller with Radio and Controller Battery installed
- VEX USB to Micro USB Cable

Once ROBOTC is installed on your computer and your VEX IQ Brain is charged it is very easy to install onto your robot's brain.

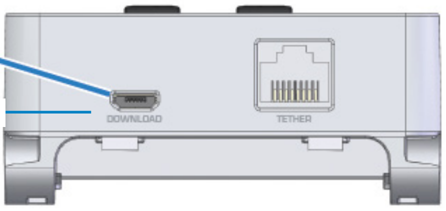
1. Plug the USB into a port on your computer
2. Plug the micro USB into your download port on the VEX IQ
3. Open the ROBOTC Graphical Language software
4. Select the Firmware Download icon and the software will install onto the VEX IQ Brain.



1. Plug the USB into the computer

Download Port

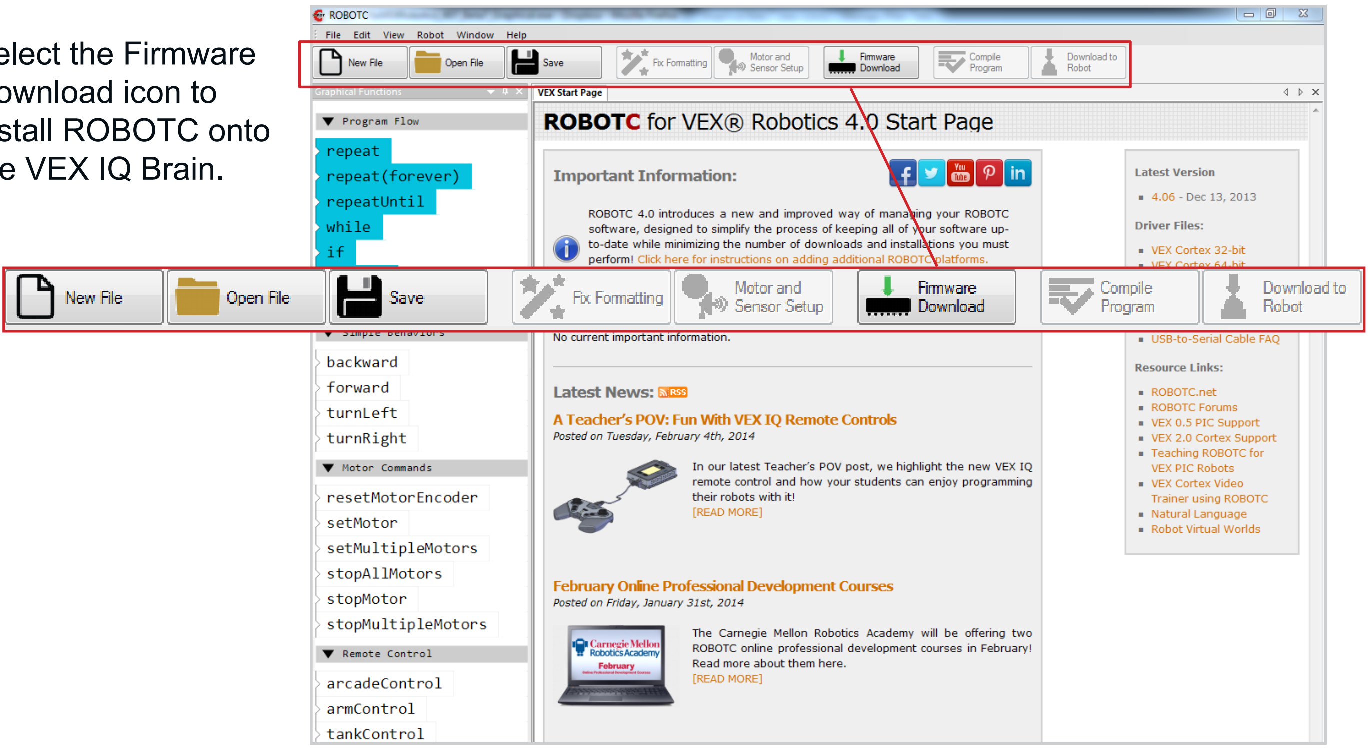
2. Plug the Micro USB to the VEX IQ



3. Open the software.

ROBOTC for VEX Robotics - Graphical

4. Select the Firmware Download icon to install ROBOTC onto the VEX IQ Brain.



ROBOTC for VEX® Robotics 4.0 Start Page

Important Information:

ROBOTC 4.0 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.

No current important information.

Latest News: RSS

A Teacher's POV: Fun With VEX IQ Remote Controls
Posted on Tuesday, February 4th, 2014

In our latest Teacher's POV post, we highlight the new VEX IQ remote control and how your students can enjoy programming their robots with it!
[READ MORE]

February Online Professional Development Courses
Posted on Friday, January 31st, 2014

The Carnegie Mellon Robotics Academy will be offering two ROBOTC online professional development courses in February! Read more about them here.
[READ MORE]

Latest Version
- 4.06 - Dec 13, 2013

Driver Files:
- VEX Cortex 32-bit
- VEX Cortex 64-bit
- USB-to-Serial Cable FAQ

Resource Links:
- ROBOTC.net
- ROBOTC Forums
- VEX 0.5 PIC Support
- VEX 2.0 Cortex Support
- Teaching ROBOTC for VEX PIC Robots
- VEX Cortex Video Trainer using ROBOTC
- Natural Language
- Robot Virtual Worlds

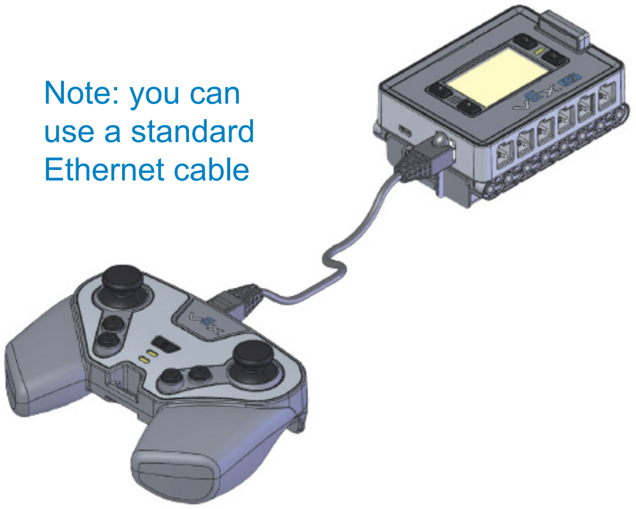
Initial Wireless Setup

Items you will need:

- Robot Brain with Radio and Robot Battery installed
- VEX Controller with Radio and Controller Battery installed
- Tether Cable P/N: 228-2786

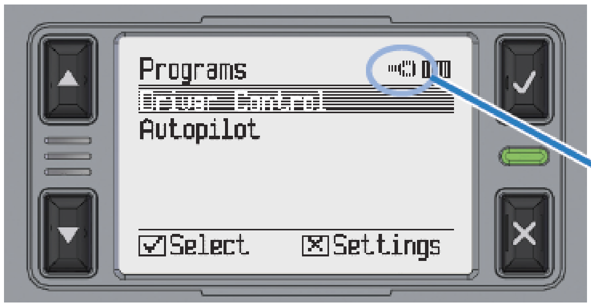
In order for the Robot Brain and Controller to communicate wirelessly, they must be paired together. Before pairing these devices together, a radio and battery must be installed into each of them. With both devices OFF, connect the Robot Brain to the Controller using the Tether Cable.

Note: you can use a standard Ethernet cable



Turn the Robot Brain ON by pressing the Check button. The Controller will automatically link and pair to the Robot Brain. The Tether Icon will appear on the Robot Brain LCD screen.

Remove the Tether cable from Robot Brain and Controller. They are now communicating wirelessly as indicated by the Radio Bar icon on the LCD screen. The Robot Brain's LED and the Controller's Power/ Link LED should be blinking green. Congratulations, your Robot Brain and Controller are now paired!



Tether Icon. See table below for explanation.

If the Robot Brain and Controller are not linked (indicated by animated "Searching" icon), turn them both OFF and repeat the process.

Icon Table

▂ ▃ ▅ ▇ 1 2 3 4 Animated icon	Searching Icon - Searching for Controller (not connected
▬◘	Tether Icon - Connected by Tether Cable
▂▃▅▇	Radio Link Icon - Connected by Radio (number of bars indicates signal strength
	No Radio installed, no Tether connected

Uploading Autonomous vs RC

There are two settings in ROBOTC Graphical Language VEX IQ Controller Mode; TeleOp - Remote Control Required and Autonomous - No Remote Control Required.

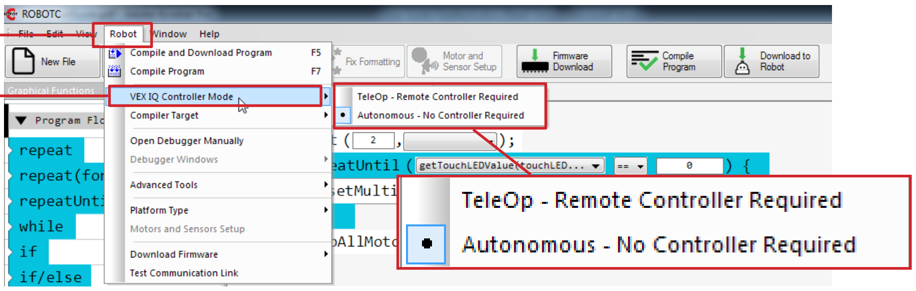
Autonomous - No Remote Control Required Mode

You would choose autonomous mode if you have written a program and you want your robot to travel around autonomously.

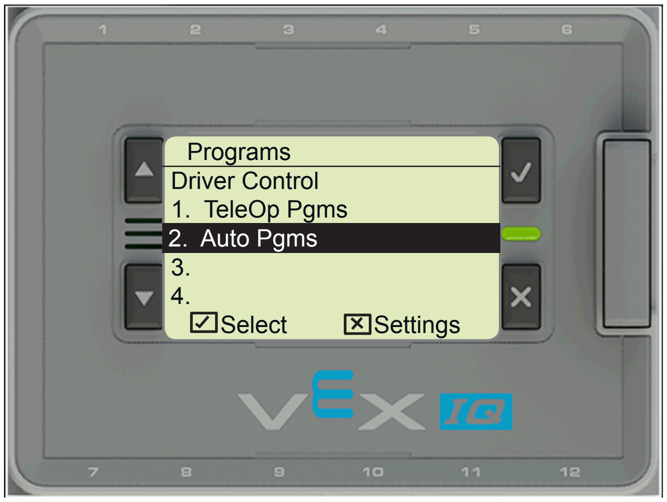
Select:

Robot

VEX IQ Controller Mode



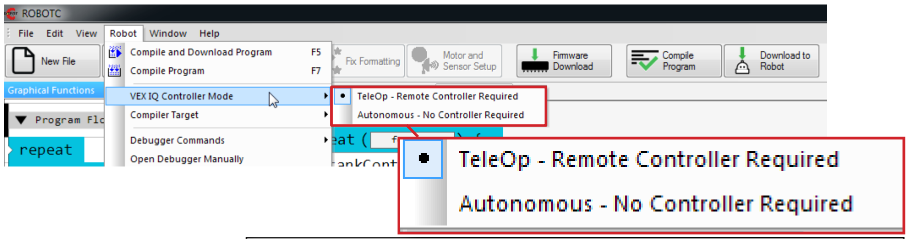
There are three steps to download and run an autonomous program, first configure your software, download your code, and then navigate to the code.



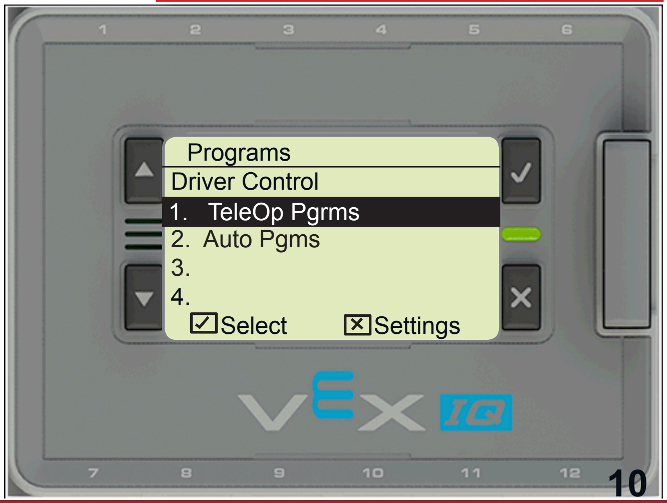
1 Select *Robot> VEX IQ Controller Mode Autonomous - Nor Controller Required*
2. Download the program to the robot
3. Navigate the VEX IQ LCD Display and select Programs>Auto Pgms and select the check on the rights side of VEX IQ LCD display.

TeleOp - Remote Control Required Mode

You would choose TeleOp mode if you are running a program that uses the VEX IQ remote control.



There are three steps to download and run an RC program, first configure your software, download your code, and then navigate to the code.

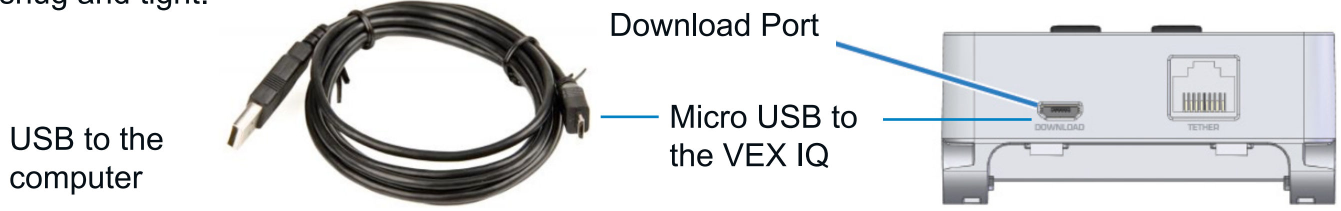


1. Select Robot > VEX IQ Controller Mode > TeleOp - Remote Control Required
2. Download the program to the robot
3. Navigate the VEX IQ LCD Display and select Programs>TeleOp Pgms and select the check on the rights side of VEX IQ LCD display.

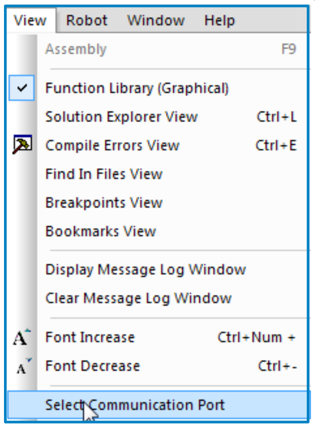
Troubleshooting

I Can't Compile and Download my program

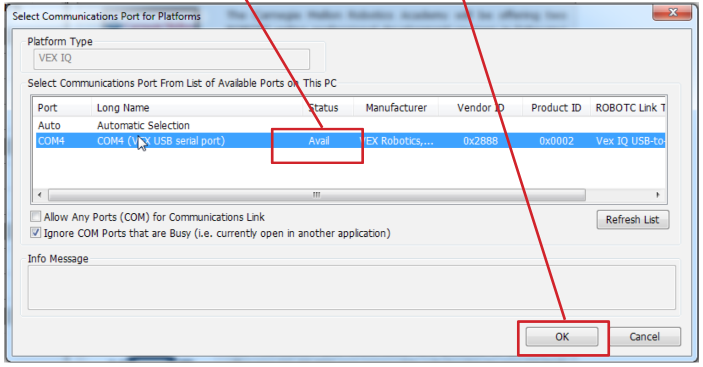
1. Unplug the Micro-USB cable from the VEX IQ brain and plug it back in. Ensure it's completely snug and tight.



2. Reset the communications port. Select View > Select Communications Port > if everything is setup and communicating correctly it should say that it is available. Select OK



Once you click "Select Communications Port" the screen on the left pops up. Select OK.

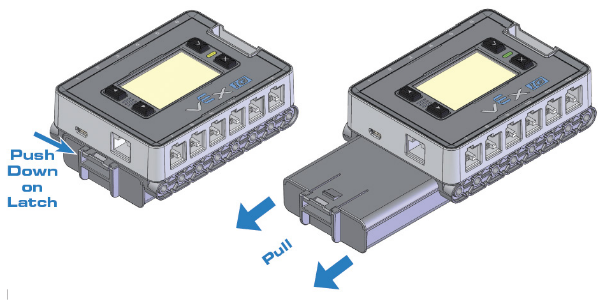

Select Communications Port for Platforms
Platform Type
VEX IQ
Select Communications Port From List of Available Ports on This PC
Refresh List
Info Message
OK
Cancel

Port	Long Name	Status	Manufacturer	Vendor ID	Product ID	ROBOTC Link T
Auto	Automatic Selection					
COM4	COM4 (VEX USB serial port)	Avail	VEX Robotics,...	0x2888	0x0002	Vex IQ USB-to

☐ Allow Any Ports (COM) for Communications Link
☑ Ignore COM Ports that are Busy (i.e. currently open in another application)

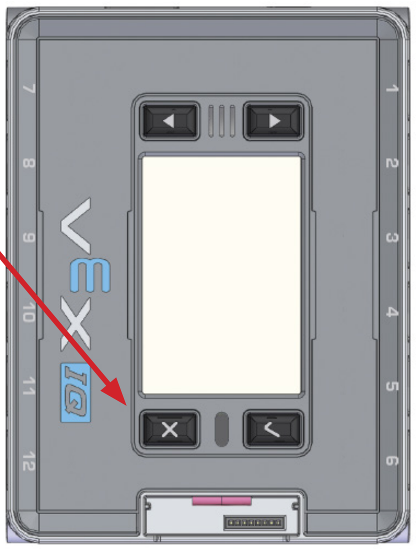
My VEX IQ is frozen

Brain Frozen?: This can happen sometimes and IFI is actively working on keeping it from happening. The VEX IQ freezes, you can recover the VEX IQ brain by unplugging the battery and plugging it back in. No need to download the firmware again.



My Motors and Sensors aren't working

1. Ensure that all cables are plugged in all of the way and give your robot a reboot by holding down the "X" button for 5 seconds.
2. Press the check mark button to turn the robot back on. The VEX IQ detects all of the devices connected only when the robot is first powered on.

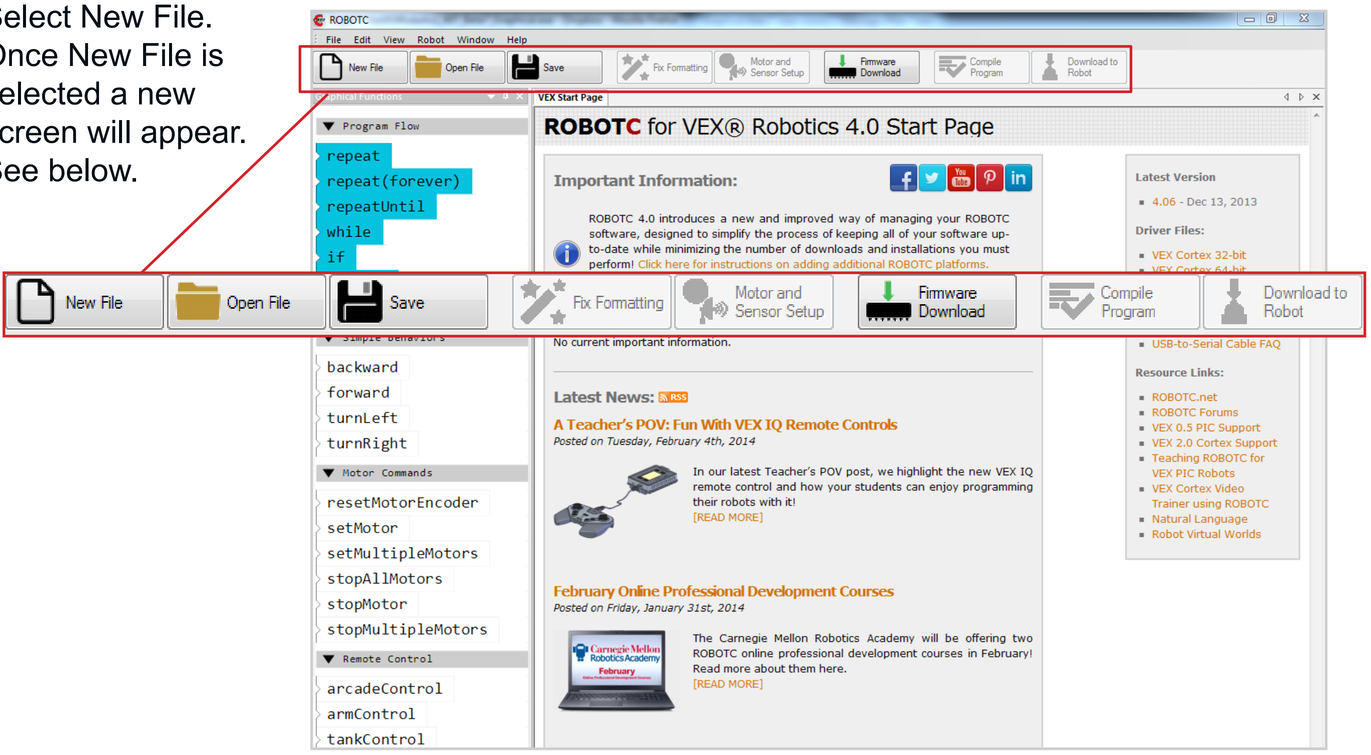


It is important to note that every time you plug a new motor or sensor into the VEX IQ controller you will need to turn the controller all the way off and then turn it back on for the controller to recognize the new motor or sensor.

How do I start a new program?

Open the software. ROBOTC for VEX Robotics - Graphical

Select New File.
Once New File is selected a new screen will appear.
See below.



ROBOTC for VEX® Robotics 4.0 Start Page

Important Information:

ROBOTC 4.0 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.

No current important information.

Latest News: RSS

A Teacher's POV: Fun With VEX IQ Remote Controls
Posted on Tuesday, February 4th, 2014

In our latest Teacher's POV post, we highlight the new VEX IQ remote control and how your students can enjoy programming their robots with it!
[READ MORE]

February Online Professional Development Courses
Posted on Friday, January 31st, 2014

The Carnegie Mellon Robotics Academy will be offering two ROBOTC online professional development courses in February! Read more about them here.
[READ MORE]

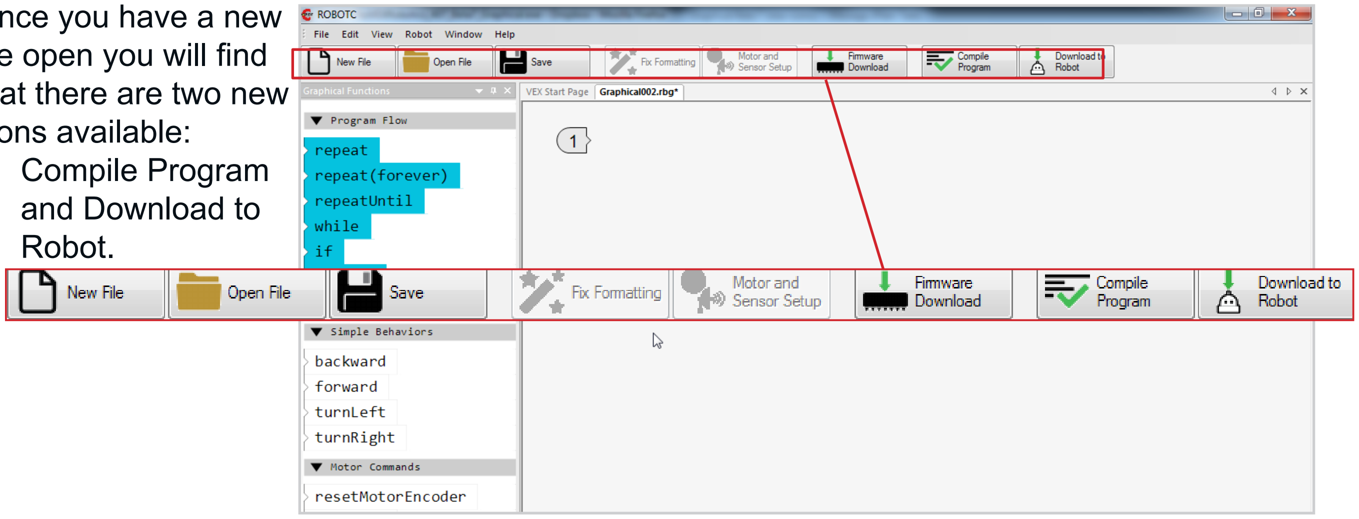
Latest Version
- 4.06 - Dec 13, 2013

Driver Files:
- VEX Cortex 32-bit
- VEX Cortex 64-bit
- USB-to-Serial Cable FAQ

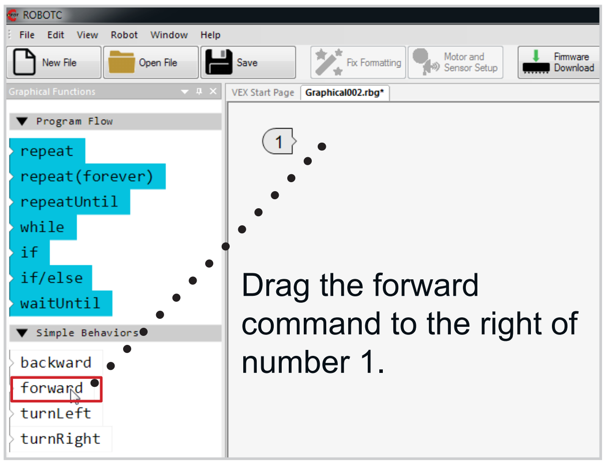
Resource Links:
- ROBOTC.net
- ROBOTC Forums
- VEX 0.5 PIC Support
- VEX 2.0 Cortex Support
- Teaching ROBOTC for VEX PIC Robots
- VEX Cortex Video Trainer using ROBOTC
- Natural Language
- Robot Virtual Worlds

Once you have a new file open you will find that there are two new icons available:

- Compile Program
- and Download to Robot.



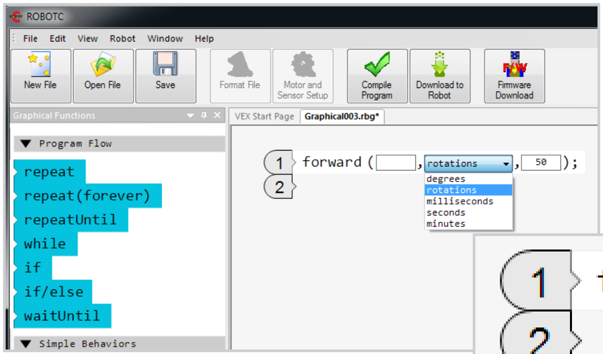
How do I write a program?



Drag the forward command to the right of number 1.

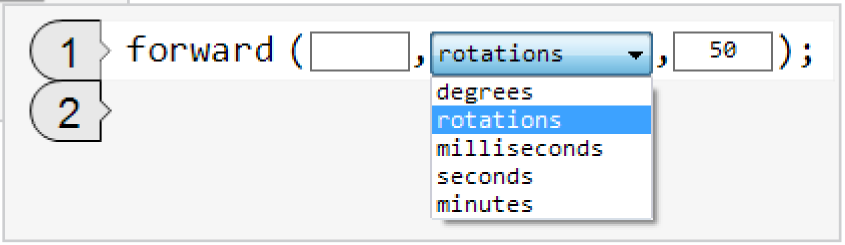
Moving Forward

In this example we will show you the simplest of all programs; programming your robot to move forward. Select "Forward" and left click with the mouse and drag the "Forward" block to the right of the "number one" start block on the screen. See below.

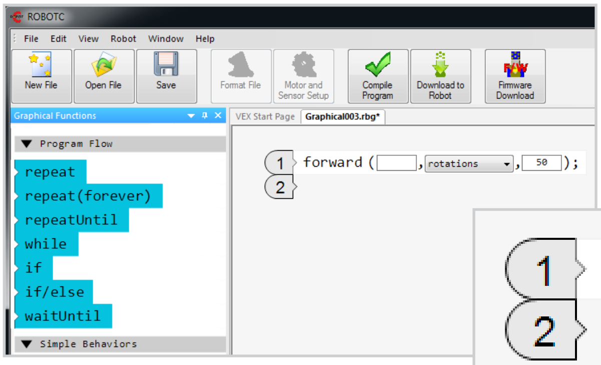


Select the Duration Type with the mouse and decide the type of unit you will use:

- Degrees
- Rotations
- or Time in milliseconds, seconds, or minutes

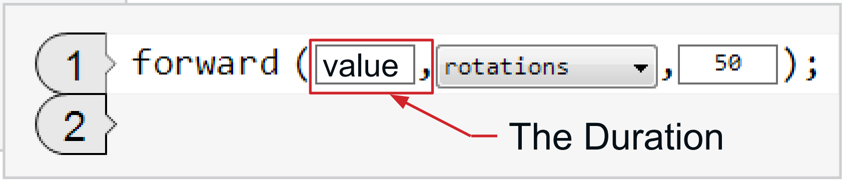


```
forward (      , rotations , 50 );
```

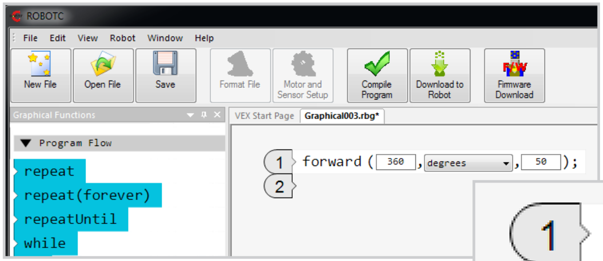


The Forward Block allows you to control three different values:

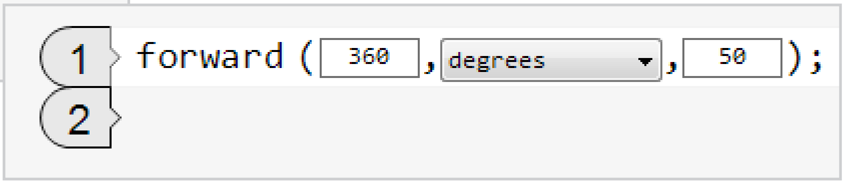
- The Duration - how many or how much
- The Duration Type - degrees, rotations, or time
- The Power Level - how fast will the robot go



```
forward ( value , rotations , 50 );
```



Set the Duration, the Duration Type, and the Power Level. In the example below the robot will travel 360 degrees at 50% power. Select compile and you will be prompted to save you file.



```
forward ( 360 , degrees , 50 );
```

How do I save/open my program?

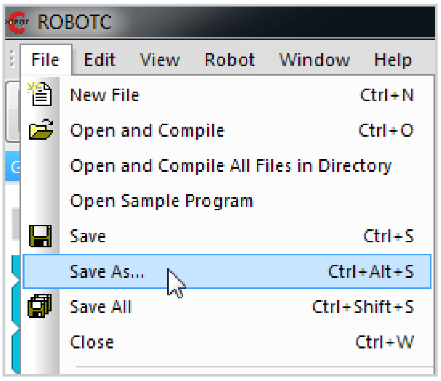
Once you've built a program you will want to save it.

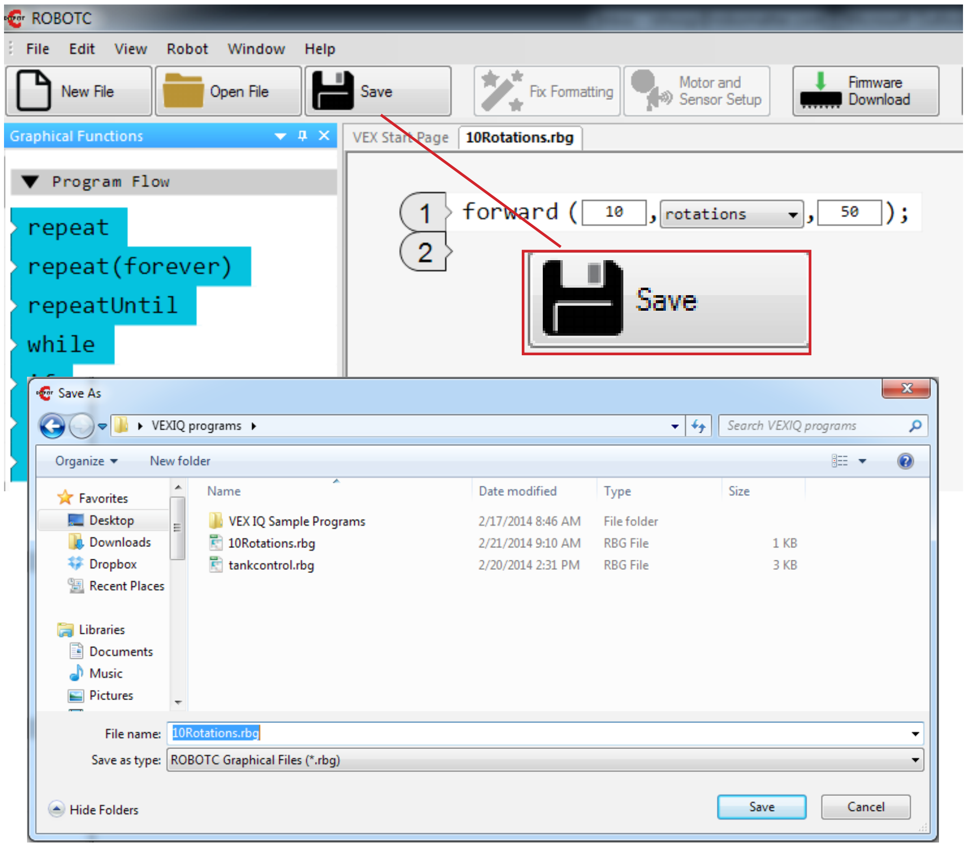
Saving Files

When you select the save icon a standard menu will pop-up allowing you to navigate to the folder that you want to save to.

Save As

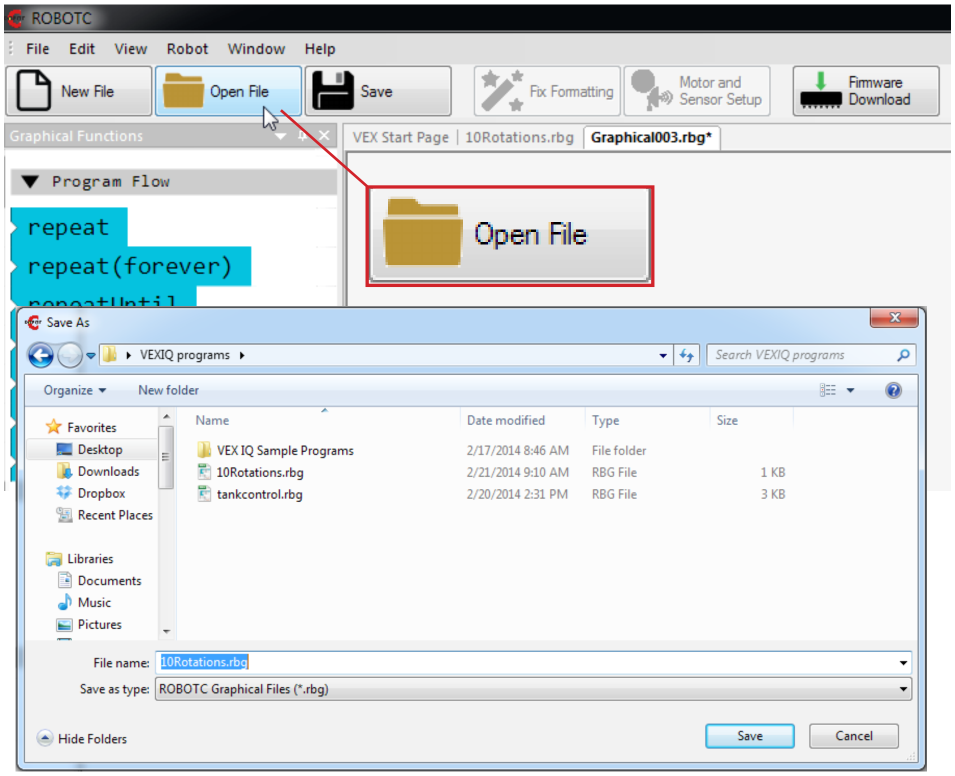
You can also use the standard Windows options such as:
File > Save As...





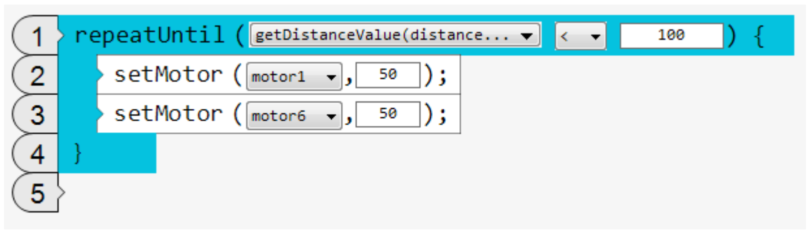
Open File

When you select the open icon a standard menu will pop-up allowing you to navigate to the folder that you want to open.



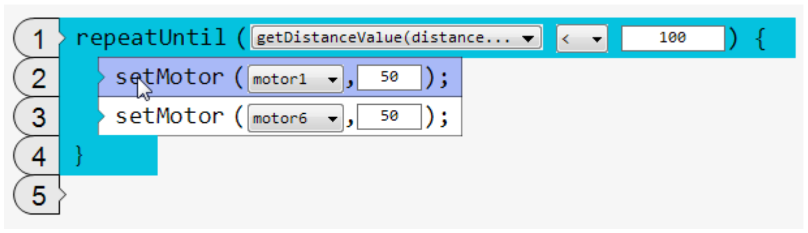
How do I edit a program?

Example Program



```
repeatUntil ( getDistanceValue(distance... < 100 ) {
    setMotor ( motor1, 50 );
    setMotor ( motor6, 50 );
}
```

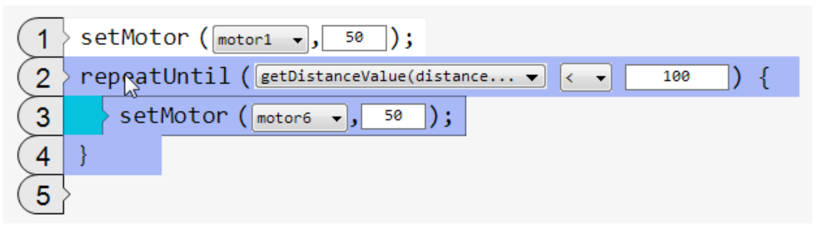
Deleting One Line



```
repeatUntil ( getDistanceValue(distance... < 100 ) {
    setMotor ( motor1, 50 );
    setMotor ( motor6, 50 );
}
```

Select the line of code that you want to delete by left clicking with the mouse. Once the code is selected use the delete key to delete the line of code.

Deleting Multiple Lines



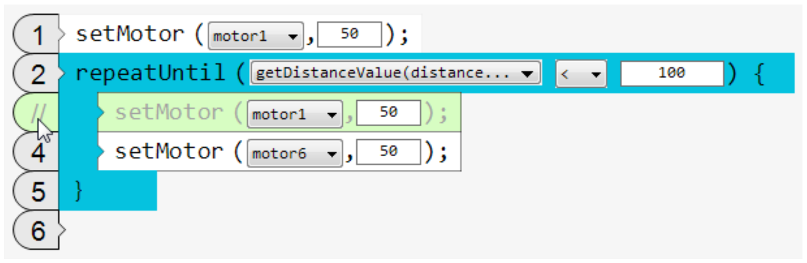
```
setMotor ( motor1, 50 );
repeatUntil ( getDistanceValue(distance... < 100 ) {
    setMotor ( motor6, 50 );
}
```

If you want to delete a loop (the blue colored command blocks, use the same process. Left click the block of code that you want to delete, it will turn purple, and select the delete key on the keyboard.

Try it and see if it works!

Commenting Out A Line

Programmers use a tool called a “comment” that allows them to ignore a line of code if they choose to. In the example below, when the program reaches line 3 it skips over it and executes the rest of the program.

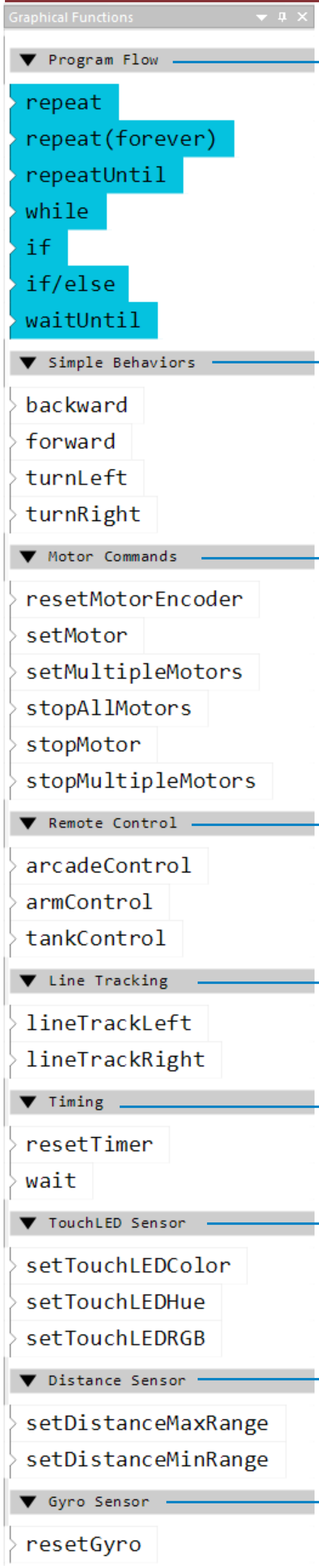


```
1  setMotor ( motor1, 50 );
2  repeatUntil ( getDistanceValue(distance... < 100 ) {
//     setMotor ( motor1, 50 );
4      setMotor ( motor6, 50 );
5  }
6
```

You can comment out a line of code by selection the number at the beginning of the line of code and then left clicking the number.

Try it and see if it works!

ROBOTC Graphical Language Commands



The Program Flow Commands are colored blue in the program and allow the robot to make decisions. To learn more about the Program Flow commands see pages 13 - 21.

The Simple Behaviors Commands allows you to quickly write code for a standard build VEX IQ robot. These commands assume that the VEX IQ's left motor is connected to port 1 and the right motor is connected to port 6, see page 22.

The Motor Commands allows the programmer to program motors setup in any configuration.

Remote Control Commands must be placed into a repeat (forever) or while (true) loop. See page xx.

The Line Track Behavior Commands allows the programmer to quickly setup line tracking behaviors for the standard VEX IQ build.

Timers - ROBOTC has several built in timers that a programmer can use for robot control and debugging purposes.

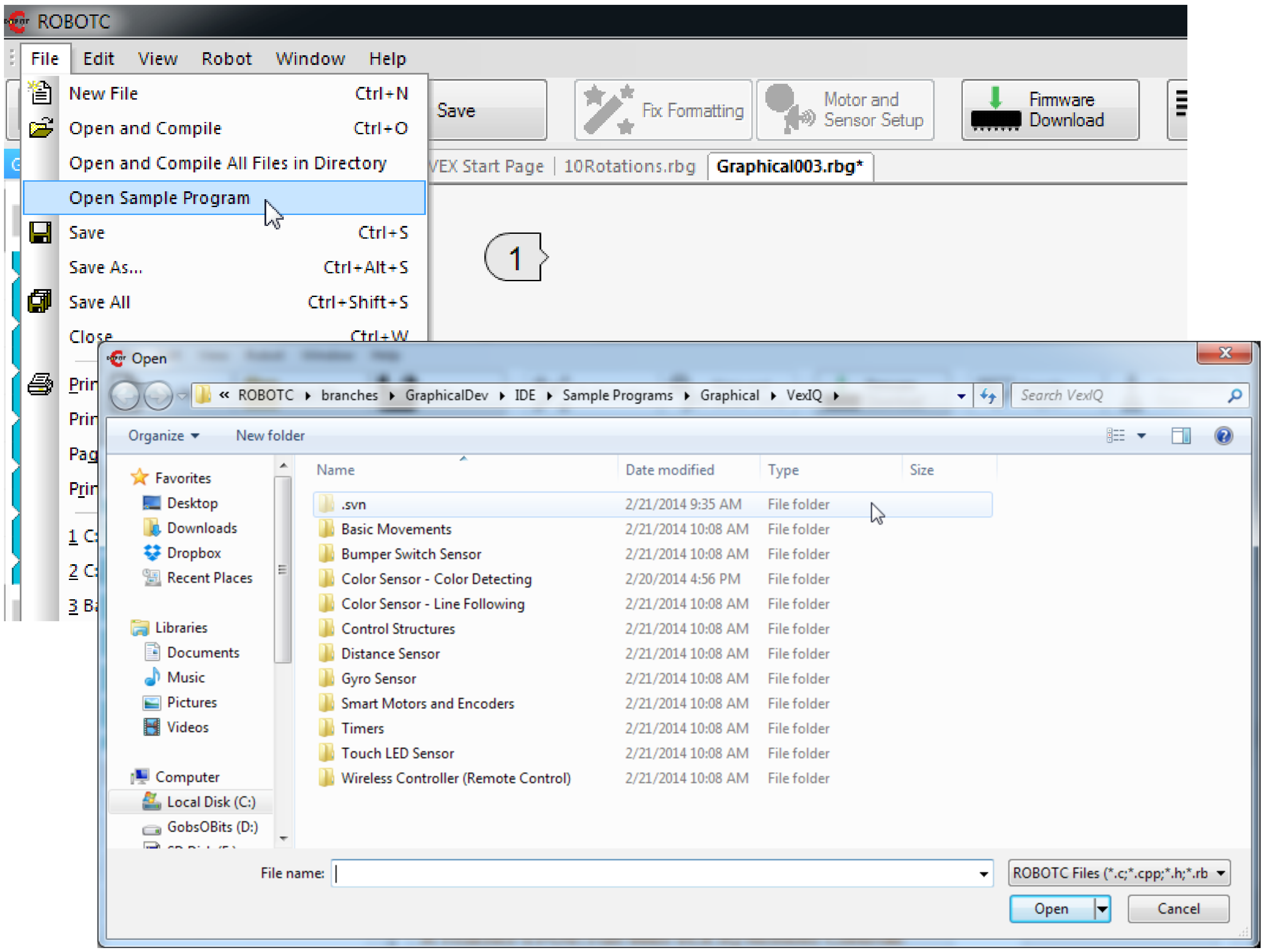
The Touch LED Sensor can be programmed to show many different colors allowing the sensor to be used to send visual signals from the robot to the operator.

The Distance Sensor is an Ultrasonic sensor that measures distance from the sensor to an object.

The resetGyro Command allows the user to reset the Gyro Sensor at any given point in time and then accurately turn their robot or robot arm a specific number of degrees.

Sample Programs

ROBOTC Graphical Language comes with lots of example programs. To access these programs go to File > Open Sample Program



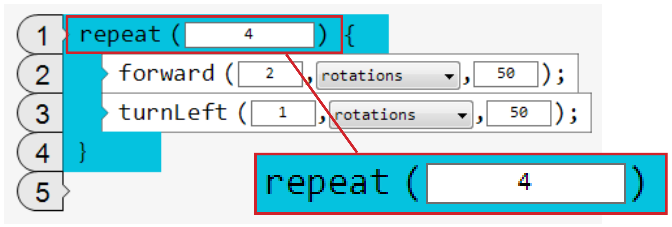
Name	Date modified	Type	Size
.svn	2/21/2014 9:35 AM	File folder	
Basic Movements	2/21/2014 10:08 AM	File folder	
Bumper Switch Sensor	2/21/2014 10:08 AM	File folder	
Color Sensor - Color Detecting	2/20/2014 4:56 PM	File folder	
Color Sensor - Line Following	2/21/2014 10:08 AM	File folder	
Control Structures	2/21/2014 10:08 AM	File folder	
Distance Sensor	2/21/2014 10:08 AM	File folder	
Gyro Sensor	2/21/2014 10:08 AM	File folder	
Smart Motors and Encoders	2/21/2014 10:08 AM	File folder	
Timers	2/21/2014 10:08 AM	File folder	
Touch LED Sensor	2/21/2014 10:08 AM	File folder	
Wireless Controller (Remote Control)	2/21/2014 10:08 AM	File folder	

The “Repeat Structure” Program Flow

The Repeat Command

The repeat command is a loop that allows you to repeat a sequence of statements (commands to the robot) for a specified number of times. In the example below, the robot will repeat these statements four times:

- Move in a Forward two rotations at 50% power
- turnLeft for one rotation at 50% power

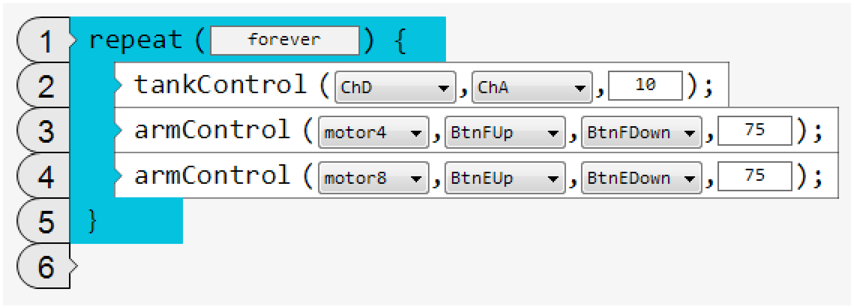


```
repeat ( 4 ) {
    forward ( 2 , rotations , 50 );
    turnLeft ( 1 , rotations , 50 );
}
```

Try it and see if it works!

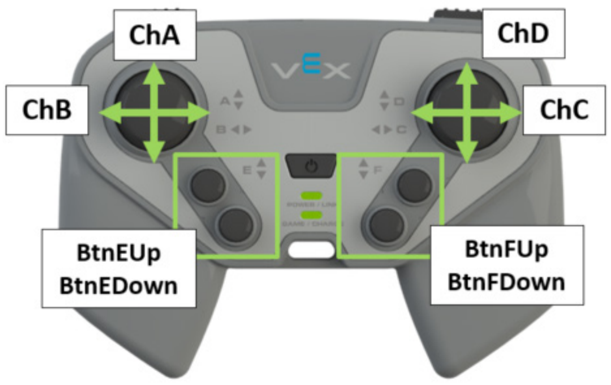
The Repeat Forever Command

The repeat command is a loop that does exactly what the name implies. It repeats everything within the Repeat command forever.



```
repeat ( forever ) {
    tankControl ( ChD , ChA , 10 );
    armControl ( motor4 , BtnFUp , BtnFDown , 75 );
    armControl ( motor8 , BtnEUp , BtnEDown , 75 );
}
```

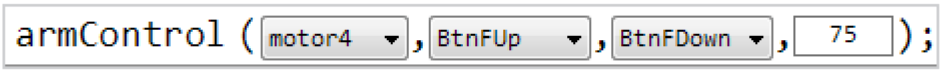
The program at the left controls a VEX IQ remote control. The “tankControl” command allows a user to map joystick ChA to motor 1 and joystick ChD to motor 6 for the standard robot build.



Push both joysticks forward and the robot moves in the forward direction, pull both backward and the robot moves in a backward direction. ChA forward and ChD backward turns right and ChD forward and ChA backward and the robot turns left.

The armControl Commands allow the programmer to set the motor that they want to control as well as the speed. There are more examples of programming the remote control in the remote control section of this guide.

Try it and see if it works!



```
armControl ( motor4 , BtnFUp , BtnFDown , 75 );
```

In the armControl command directly above, motor 4 is selected, BtnFUP sends a positive 75% power to the motor, BtnFDown sends a negative 75% power to the motor, the power level is set in the last box.

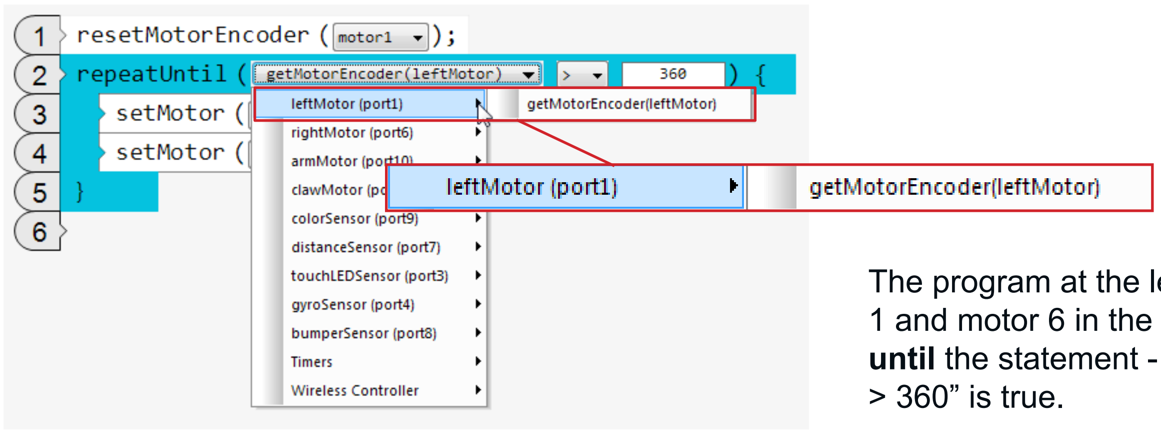
The “Repeat Structure” Program Flow

Repeat Until getMotorEncoderValue

The VEX IQ is equipped with SmartMotors that have built in encoders. It is always a good practice to reset the value of the encoder to zero when using the value of the encoder. Line one resets the value of the motorEncoder in port one to zero.

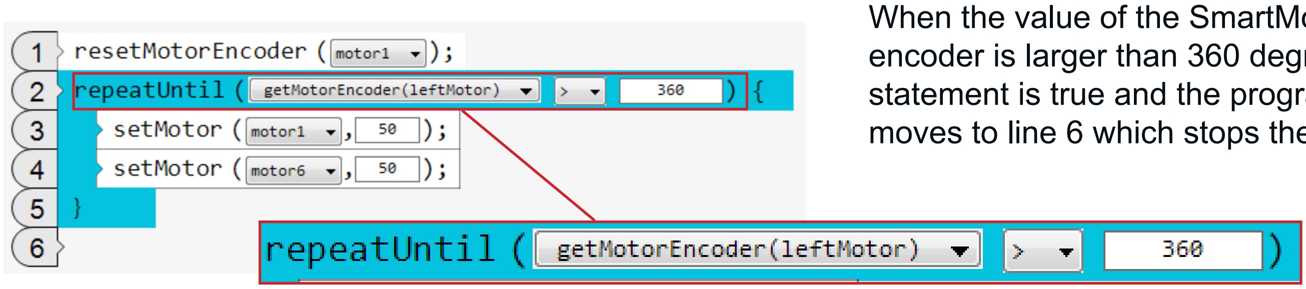
The VEX IQ SmartMotor

The Repeat Until getMotorEncoderValue command is a powerful command that enables you to use the value of the“getMotorEncoder” function to help your robot make a decision. Select the down arrow on the right of the first box and a dropdown list appears. Select getMotorEncoder (leftMotor).

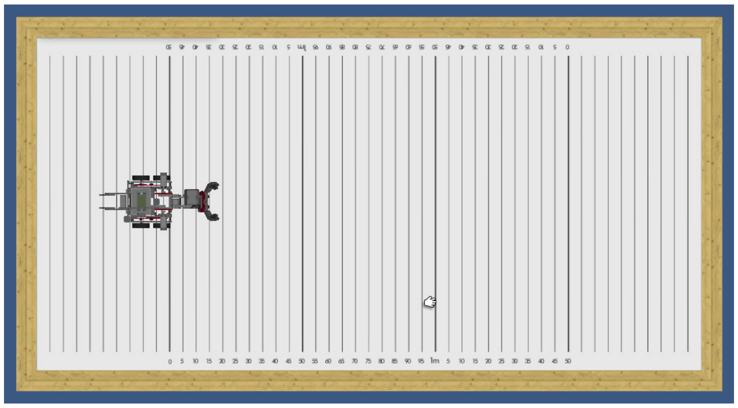

```
1  resetMotorEncoder ( motor1 );
2  repeatUntil ( getMotorEncoder(leftMotor) > 360 ) {
3     setMotor (
4     setMotor (
5  }
6
```

The program at the left will move motor 1 and motor 6 in the forward direction **until** the statement - “getMotorEncoder > 360” is true.

When the value of the SmartMotor’s encoder is larger than 360 degrees the statement is true and the program flow moves to line 6 which stops the robot.


```
1  resetMotorEncoder ( motor1 );
2  repeatUntil ( getMotorEncoder(leftMotor) > 360 ) {
3     setMotor ( motor1 , 50 );
4     setMotor ( motor6 , 50 );
5  }
6
```
```
repeatUntil ( getMotorEncoder(leftMotor) > 360 )
```

Program your robot to travel a specific distance.



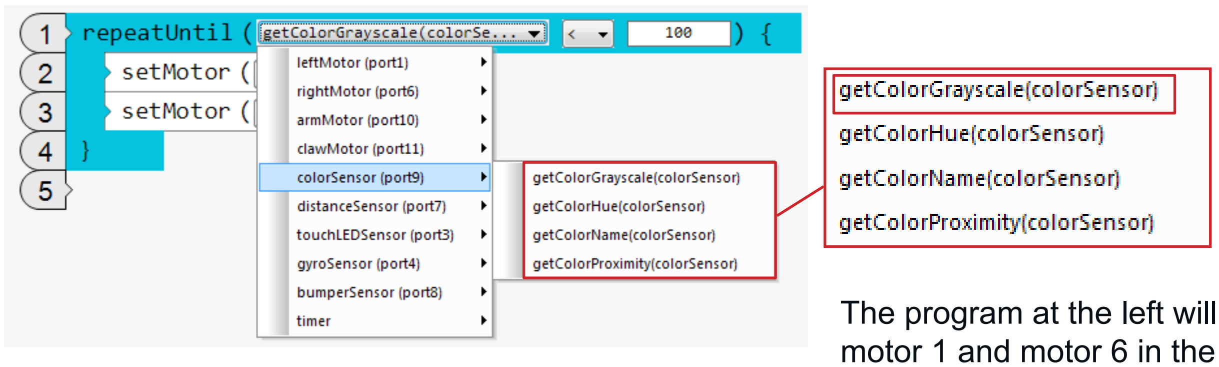
Try it and see if it works!

The “Repeat Structure” Program Flow

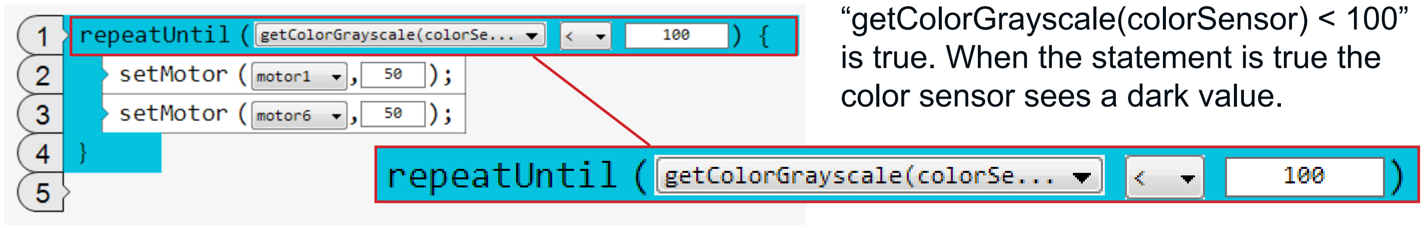
Repeat Until “getColorSensorValue”

The Repeat Until getColorSensorValue command enables you to use the value of the“getColor” function to help your robot make a decision. The values of the getColorGrayscale return values between 0 and 400. Dark objects return lower values and light objects return higher values. The program below controls the robot to travel forward until the color sensor (facing the ground) sees dark.

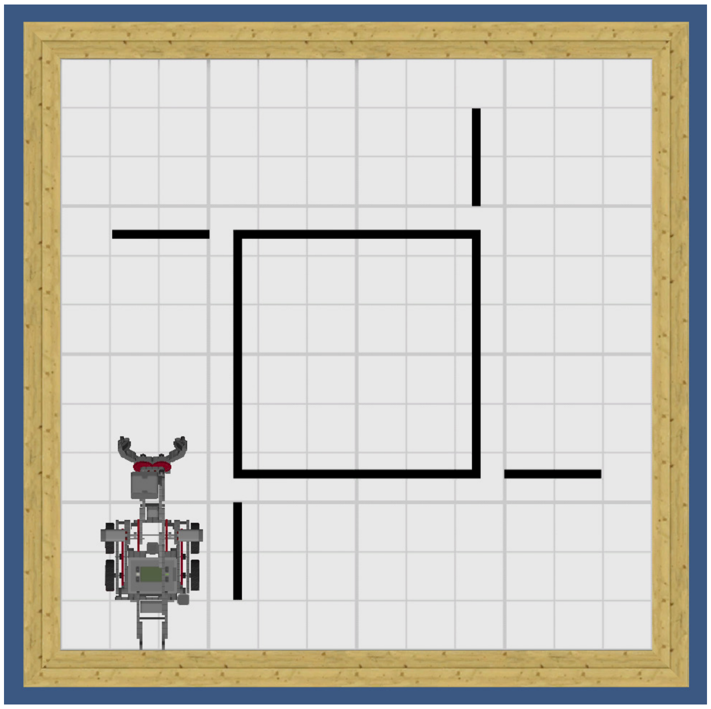
The VEX IQ Color Sensor


```
repeatUntil ( getColorGrayscale(colorSe... < 100 ) {
  setMotor (
  setMotor (
}
```

The program at the left will move motor 1 and motor 6 in the forward direction **until** the statement “getColorGrayscale(colorSensor) < 100” is true. When the statement is true the color sensor sees a dark value.


```
repeatUntil ( getColorGrayscale(colorSe... < 100 ) {
  setMotor ( motor1 , 50 );
  setMotor ( motor6 , 50 );
}
```

Use feedback from a color sensor to move forward until your robot sees dark.

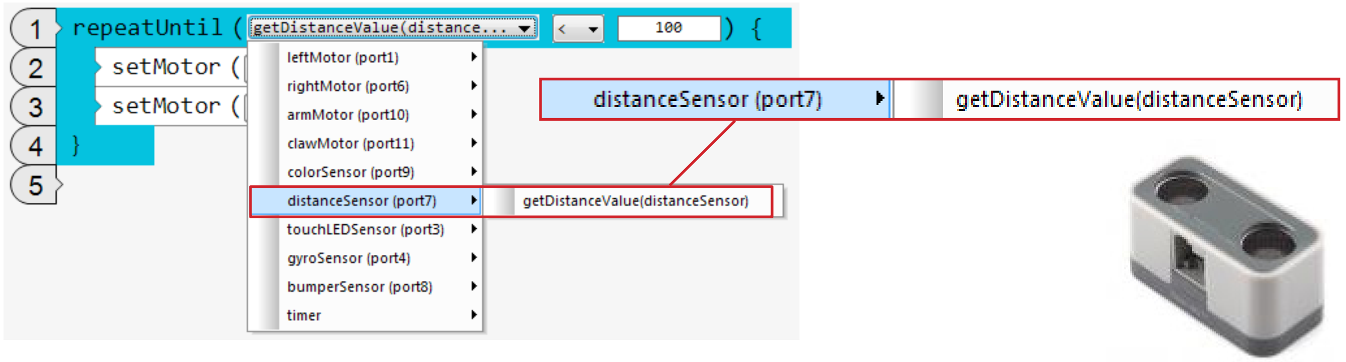


Try it and see if it works!

The “Repeat Structure” Program Flow

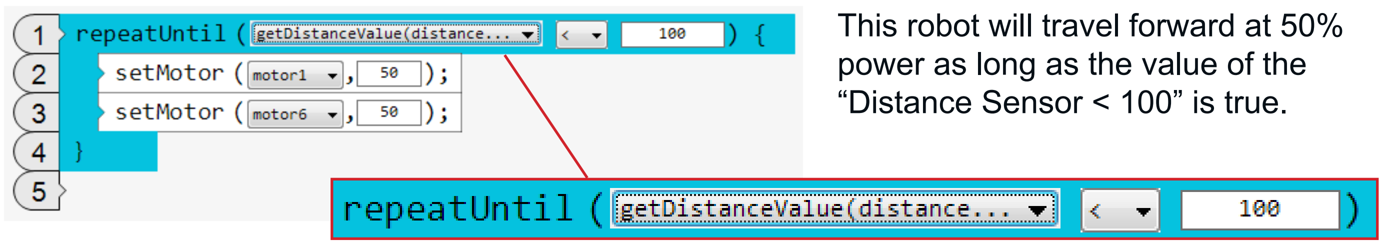
Repeat Until “getDistanceValue”

The Repeat Until getDistanceValue command enables you to use the value of the distanceSensor (a sonar sensor) to control your robot.



```
repeatUntil ( getDistanceValue(distance... < 100 ) {
    setMotor (
    setMotor (
}
```

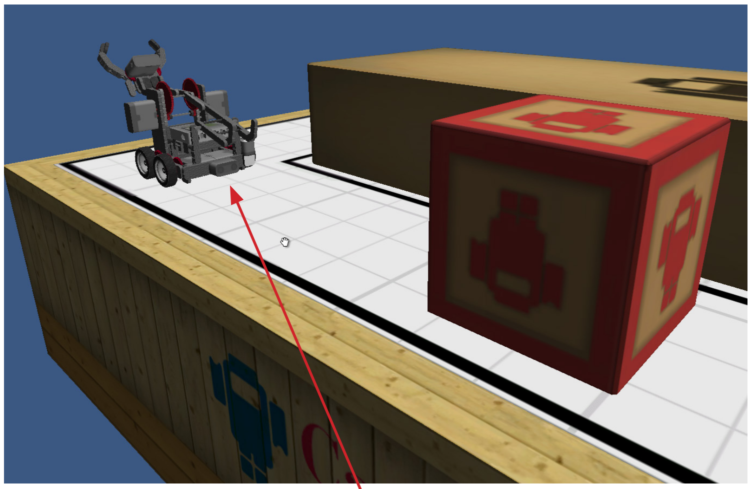
The VEX IQ Distance Sensor



```
repeatUntil ( getDistanceValue(distance... < 100 ) {
    setMotor ( motor1 , 50 );
    setMotor ( motor6 , 50 );
}
```

```
repeatUntil ( getDistanceValue(distance... < 100 )
```

This robot will travel forward at 50% power as long as the value of the “Distance Sensor < 100” is true.



Distance Sensor facing forward

The distance sensor will return the value of the closest object that is within it’s currently specified range. Values returned by the distance sensor are in millimeters. The default “range” is between 60mm-4000mm.

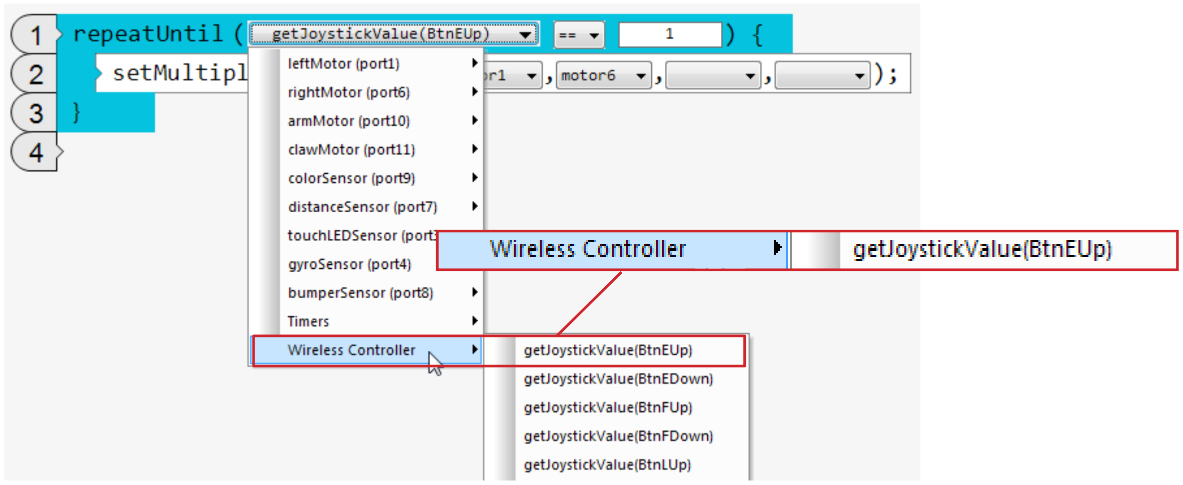
Note: If the distance sensor cannot detect an object, it will return the maximum distance value possible (26214mm).

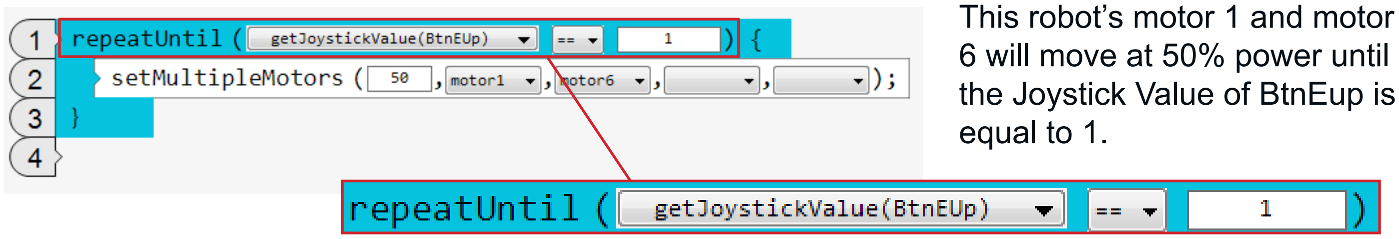
Try it and see if it works!

The “Repeat Structure” Program Flow

Repeat Until “getJoystick Value” Command

The Repeat Until getJoystickValue command enables you to values from the buttons on the Joystick to control actions on the robot



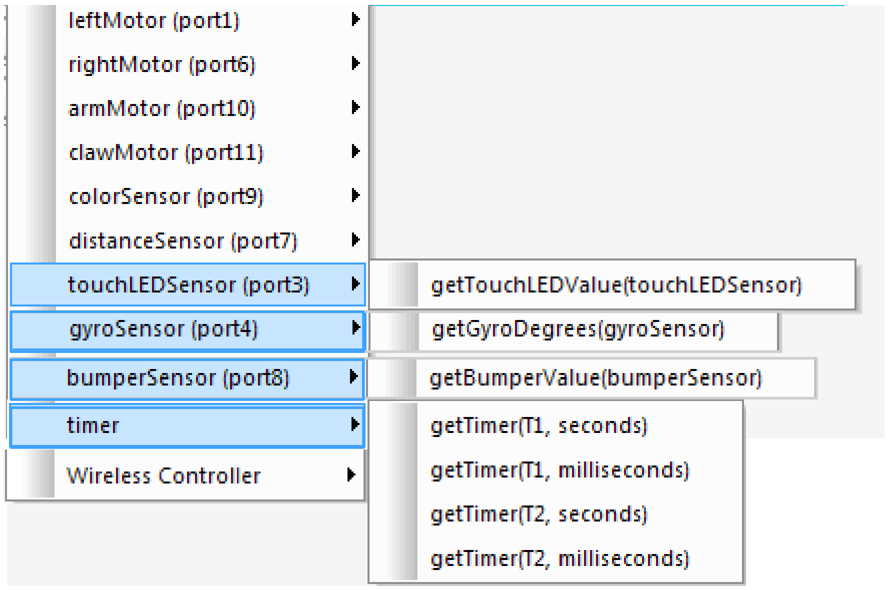


```
repeatUntil ( getJoystickValue(BtnEUp) == 1 ) {
  setMultipleMotors ( 50 , motor1 , motor6 , , );
}
```

This robot’s motor 1 and motor 6 will move at 50% power until the Joystick Value of BtnEup is equal to 1.

Try it and see if it works!

Repeat Until “getTouchLEDSensorValue”, “getGyroSensorValue”, “getBumperSensorValue”, and “getTimerValue” Commands



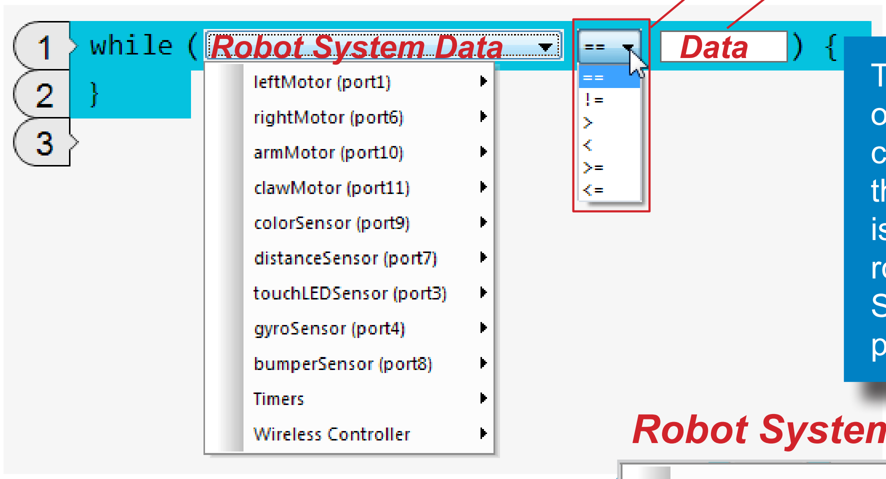
All of the “getSensorValue” commands use the same format as the “repeat until getDistanceValue command” directly above. You robot consists of smart systems (motors, sensors, and remote controls) that are monitored by the robot’s central processor (the brain) and that data is used to make decisions. See page 16 to learn more about the robot’s systems and data.

Experiment and see if this makes sense!

The “While Structure” Program Flow

The While Loop

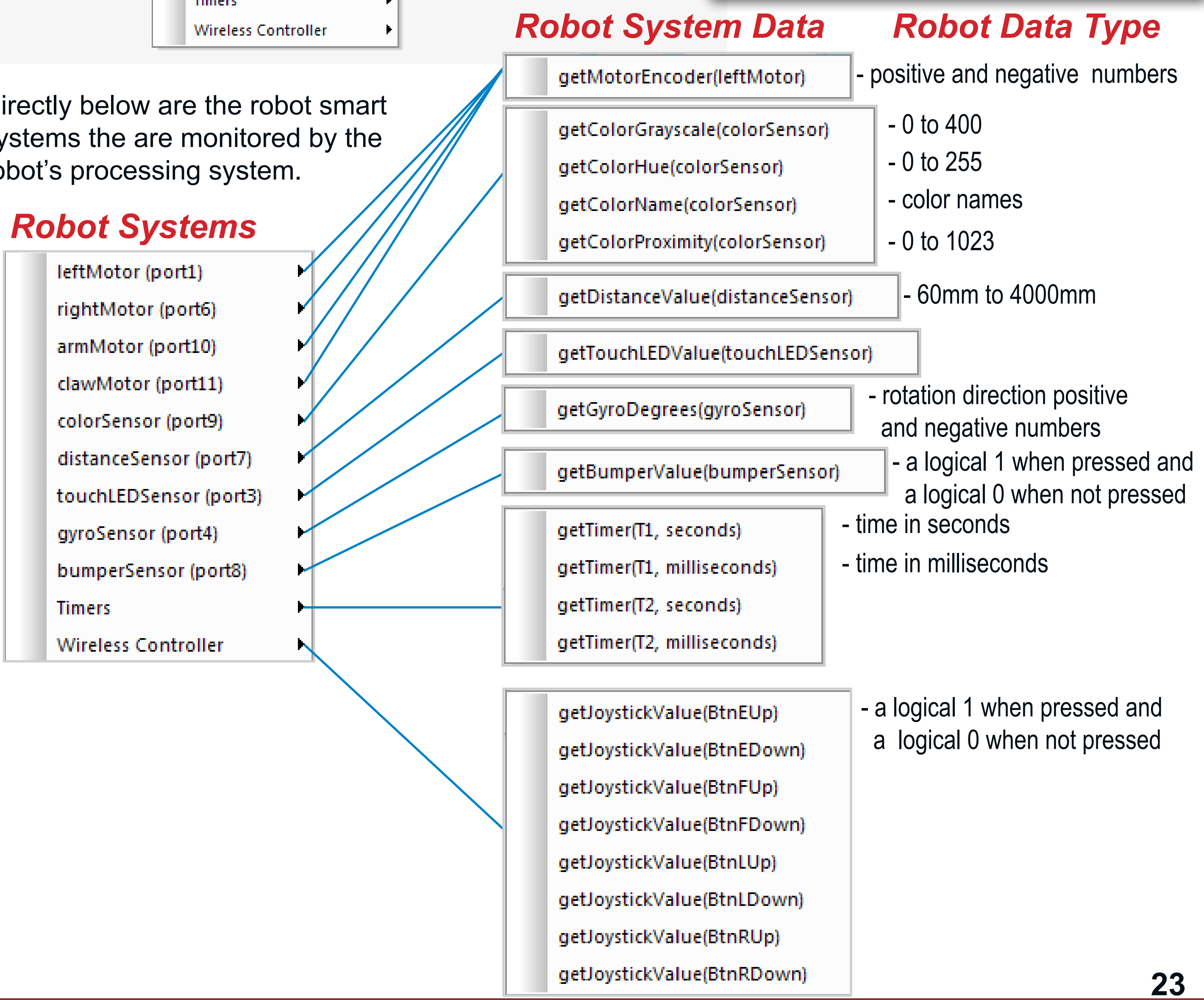
The While Loop is able to access information from all motor ports, sensors, and the joystick to control program flow.



```
while ( Robot System Data == Data ) {
}
```

The while loop uses conditional operators like >, <, ==, or != to compare the robot’s system data to the user set data. This comparison is either true of false allowing the robot to make a logic based decision. See examples on the next couple of pages.

Directly below are the robot smart systems the are monitored by the robot’s processing system.

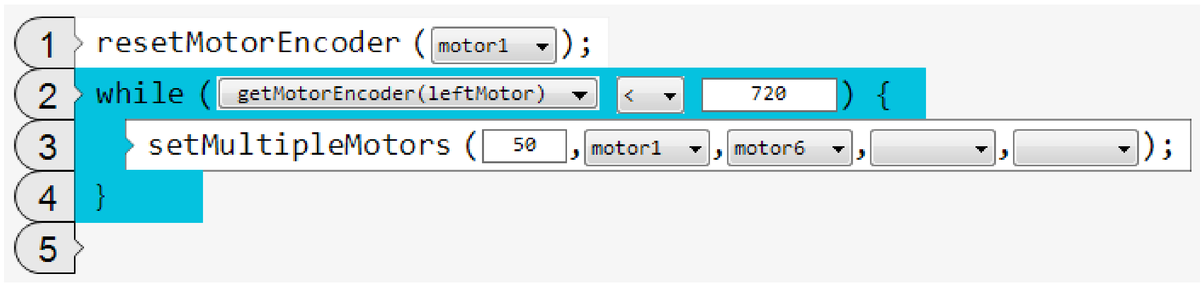


Robot Systems	Robot System Data	Robot Data Type
leftMotor (port1)	getMotorEncoder(leftMotor)	- positive and negative numbers
rightMotor (port6)		
armMotor (port10)		
clawMotor (port11)		
colorSensor (port9)	getColorGrayscale(colorSensor)	- 0 to 400
	getColorHue(colorSensor)	- 0 to 255
	getColorName(colorSensor)	- color names
	getColorProximity(colorSensor)	- 0 to 1023
distanceSensor (port7)	getDistanceValue(distanceSensor)	- 60mm to 4000mm
touchLEDSensor (port3)	getTouchLEDValue(touchLEDSensor)	
gyroSensor (port4)	getGyroDegrees(gyroSensor)	- rotation direction positive and negative numbers
bumperSensor (port8)	getBumperValue(bumperSensor)	- a logical 1 when pressed and a logical 0 when not pressed
Timers	getTimer(T1, seconds)	- time in seconds
	getTimer(T1, milliseconds)	- time in milliseconds
	getTimer(T2, seconds)	
	getTimer(T2, milliseconds)	
Wireless Controller	getJoystickValue(BtnEUp)	- a logical 1 when pressed and a logical 0 when not pressed
	getJoystickValue(BtnEDown)	
	getJoystickValue(BtnFUp)	
	getJoystickValue(BtnFDown)	
	getJoystickValue(BtnLUp)	
	getJoystickValue(BtnLDown)	
	getJoystickValue(BtnRUp)	
	getJoystickValue(BtnRDown)	

The “While Structure” Program Flow

While Smart Motor Loop (move forward for a distance)

The VEX IQ smartmotor allows very precise movements. Smartmotors begin counting as soon as they are turned and so it is a good practice to always reset your encoder before using it; line one resets motor encoder 1. Then, “while the encoder on motor 1 is less than 720 degrees (three rotations), motor 1 and motor 6 will move in the forward direction at 50% power.



```
1  resetMotorEncoder (motor1);
2  while (getMotorEncoder(leftMotor) < 720) {
3      setMultipleMotors (50, motor1, motor6, , );
4  }
5
```

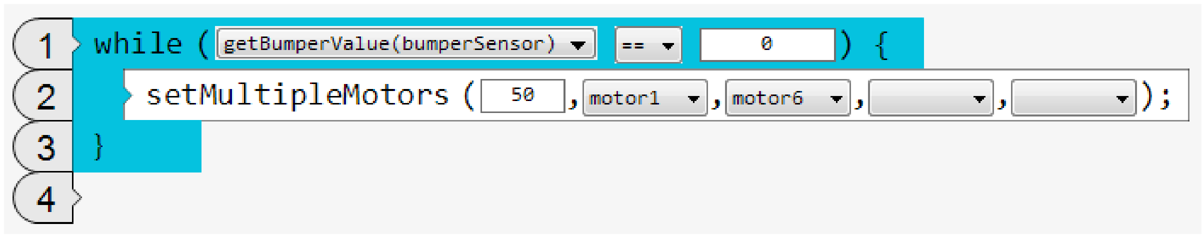
Try it and see if it works!

The VEX IQ Smart Motor

While bumperSensor Loop (move forward until touch)

The bumperSensor returns a value of “0” when it is not pressed and “1” if it is pressed. In the program directly below “while the value of the BumperSensor is equal to 0” motor 1 and motor 6 will move in the forward direction at 50% power.

The VEX IQ Bumper Sensor



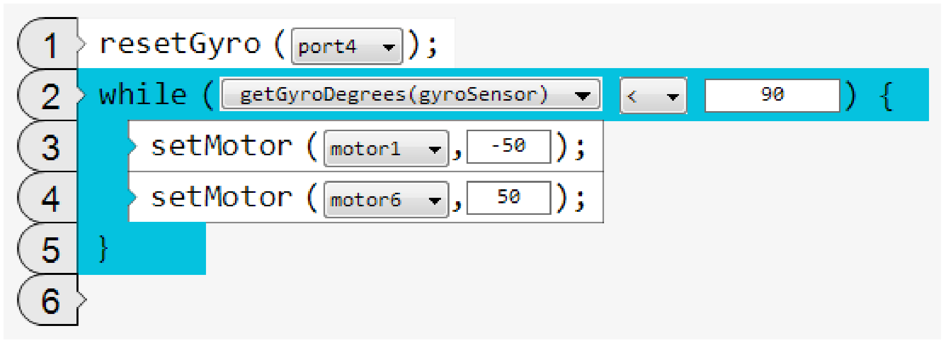
```
while (getBumperValue(bumperSensor) == 0) {
    setMultipleMotors (50, motor1, motor6, , );
}
```

Try it and see if it works!

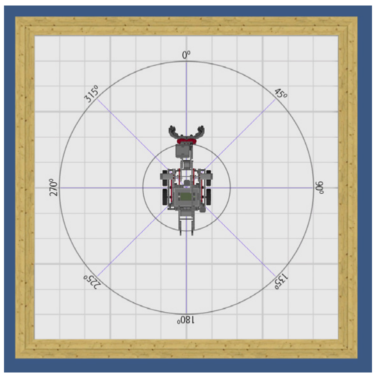
While Gyro Loop (turn using the gyro sensor)

The gyroSensor returns values that are negative or positive whole numbers (-90, 360, -270, etc.). In the example below we begin by resetting the value of the gyro sensor to 0, then “while the value of gyroSensor is less than 90 degrees” motor 1 will turn backward and motor 6 will turn forward. Program your robot to turn 90 degrees.

The VEX IQ Gyro Sensor



```
resetGyro (port4);
while (getGyroDegrees(gyroSensor) < 90) {
    setMotor (motor1, -50);
    setMotor (motor6, 50);
}
```



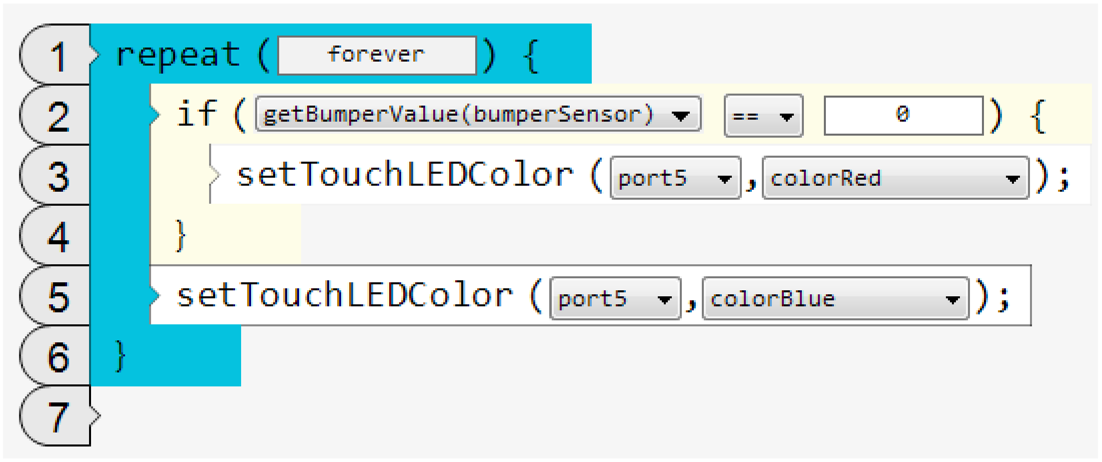
Try it and see if it works!

The “If Structure” Program Flow

The if Structure

The “If Structure” checks a the condition of the statement one time and then moves along in the program. The “if” and “if/else” programming structures are typically found embedded within a Looping Structure (like a repeat(forever) or while loop). In the example below, the if structure is embedded within a “repeat (forever) structure. If the bumperSensor is equal to 0 (not pressed) the TouchLED connected to port 5 will set to the colorRed, if it is not pressed, it will be set to the colorBlue.

The VEX IQ TouchLED Sensor



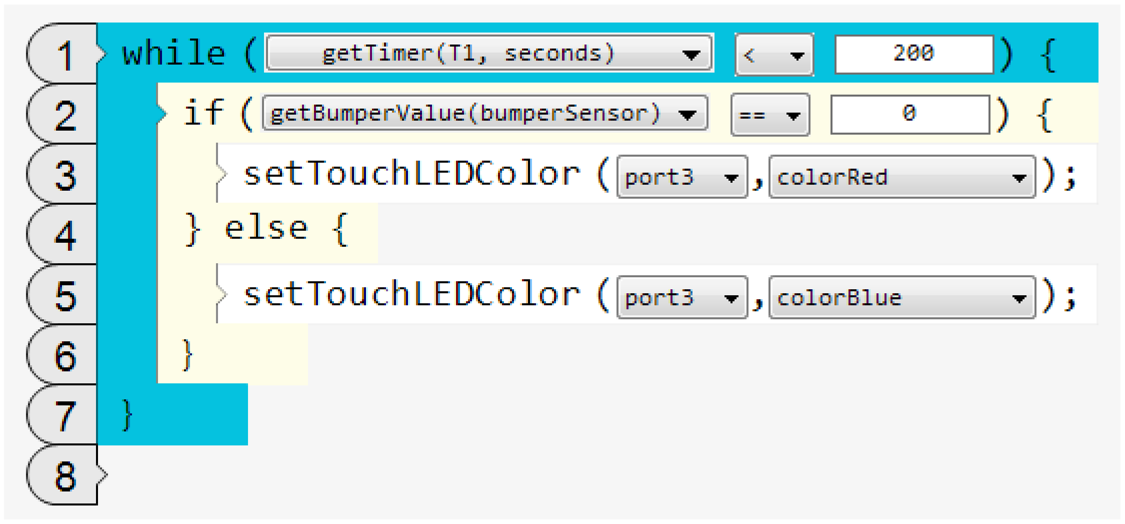
```
repeat ( forever ) {
    if ( getBumperValue(bumperSensor) == 0 ) {
        setTouchLEDColor ( port5 , colorRed );
    }
    setTouchLEDColor ( port5 , colorBlue );
}
```

The VEX IQ Bumper Sensor

Try it and see if it works!

The if/else Structure

The example below accomplishes the same behavior as shown above. In this example, while Timer1 is less than 20 seconds if the bumpSensor is pressed the TouchLED in port 3 will turn Red, if it is not pressed the Touch LED in port 3 will turn Blue.



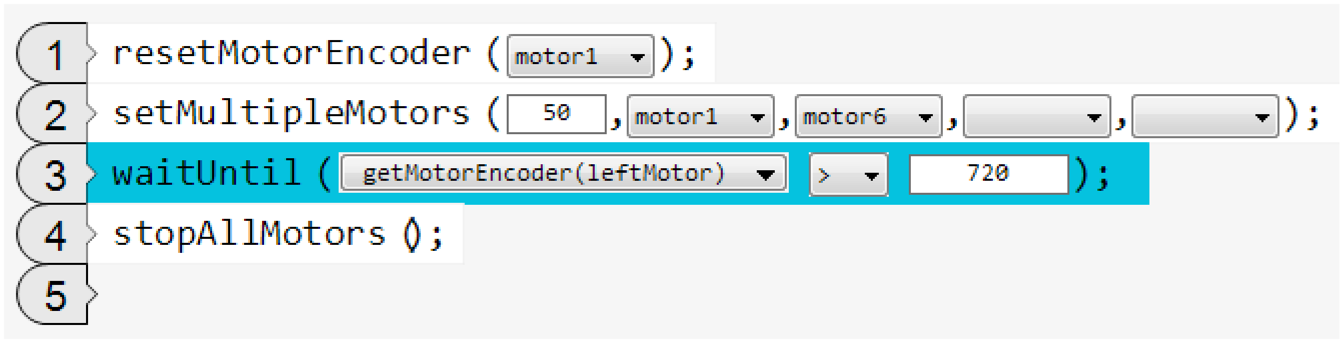
```
while ( getTimer(T1, seconds) < 200 ) {
    if ( getBumperValue(bumperSensor) == 0 ) {
        setTouchLEDColor ( port3 , colorRed );
    } else {
        setTouchLEDColor ( port3 , colorBlue );
    }
}
```

Try it and see if it works!

The “waitUntil Structure” Program Flow

The waitUntil Structure

The waitUntil Structure uses an “idle loop” to control programming flow. An idle loop stops program flow at that point in the code and waits until that condition is true. The condition in this examples is that the “leftMotorEncoder is greater than 720 degrees. The program begins by resetting the encoders, then turns on the motors, and then waits for the condition to be true, then stops.

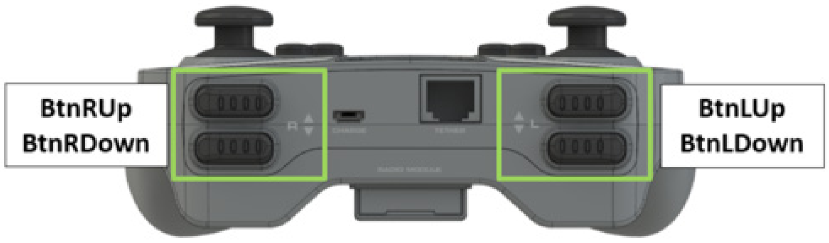


```
resetMotorEncoder (motor1);
setMultipleMotors (50, motor1, motor6, , );
waitUntil (getMotorEncoder(leftMotor) > 720);
stopAllMotors ();
```

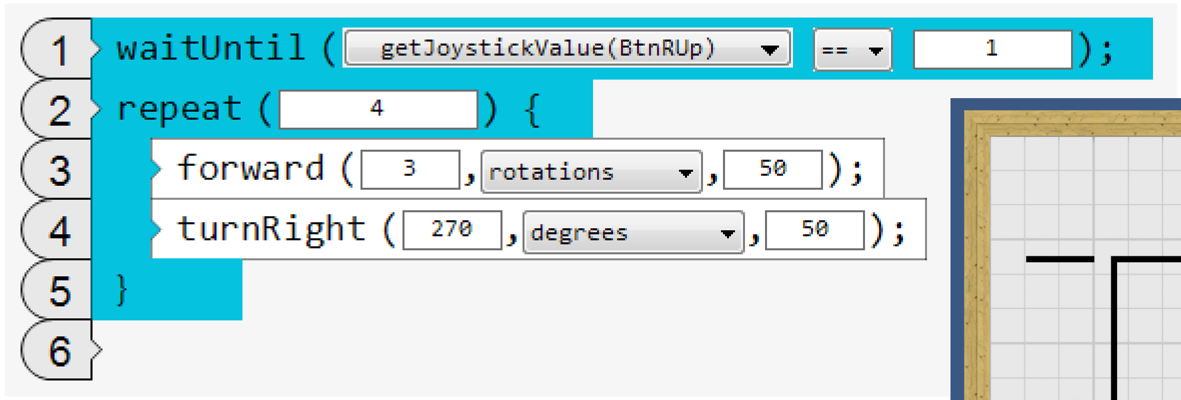
Try it and see if it works!

waitUntil BtnRUp Example

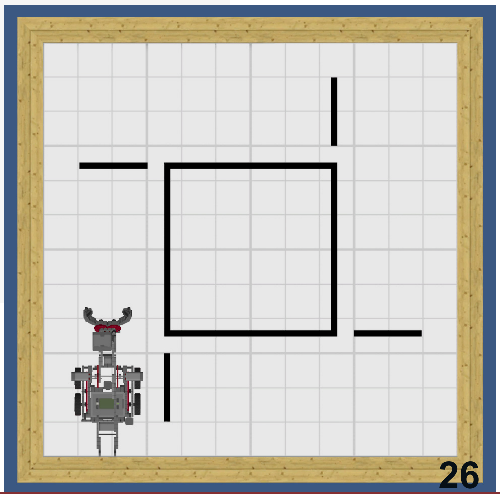
The picture at the right shows the top view of a VEX IQ remote control. In the program below we will use the remote control to start the program.



VEX IQ remote control buttons return a value of 1 if they are pressed and 0 if they are not pressed. In the program below the program will “waitUntil” BtnRUp is pressed. This program then repeats the forward and turnRight behaviors four times. If the numbers are corrected the robot will travel in a square.



```
waitUntil (getJoystickValue(BtnRUp) == 1);
repeat (4) {
    forward (3, rotations, 50);
    turnRight (270, degrees, 50);
}
```



Try it and see if it works!

Programming Simple Behaviors

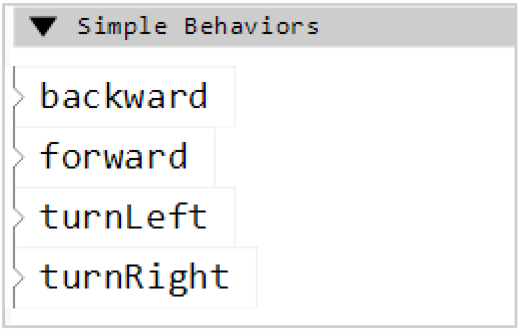
Simple Behaviors

ROBOTC Graphical Language Simple Behaviors assume that the robot has the following motor configuration:

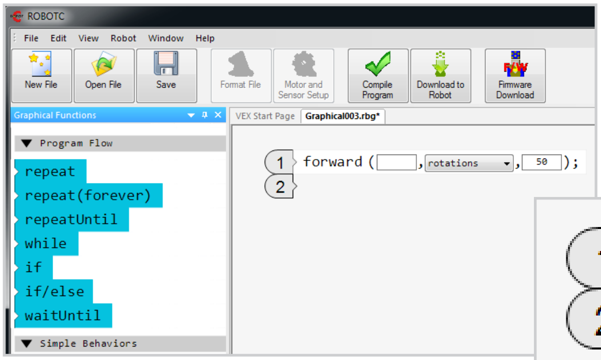
- Left Motor - port 1
- Right Motor - port 6

The following features are programmable:

- The Duration Value
- The Duration Type (degrees, rotations, or time)
- and the motor power level (speed).

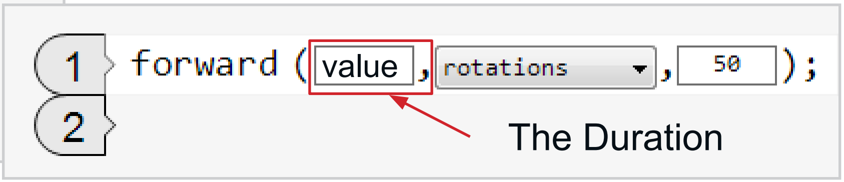


The Duration Value



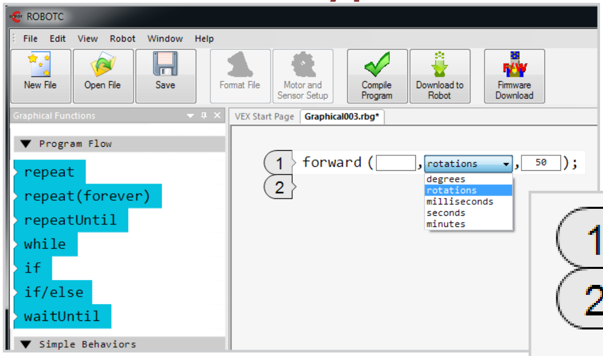
The Forward Block allows you to control three different values:

- The Duration - how many or how much
- The Duration Type - degrees, rotations, or time
- The Power Level - how fast will the robot go



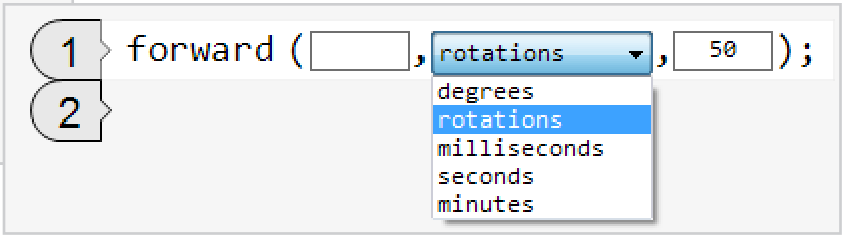
```
forward ( value , rotations , 50 );
```

The Duration Type



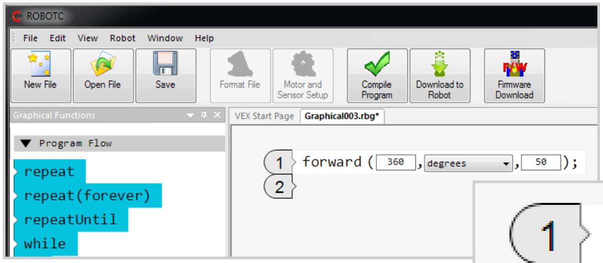
Select the Duration Type with the mouse and a drop-down list appears allowing you to select:

- Degrees
- Rotations
- or Time in milliseconds, seconds, or minutes

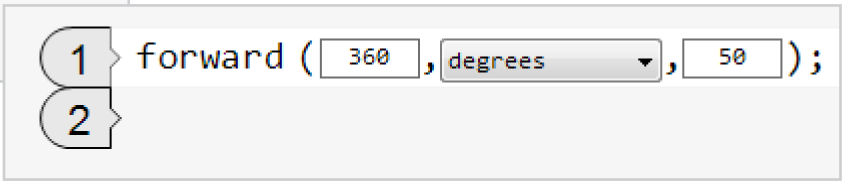


```
forward ( , rotations , 50 );
```

The Power Level



Set the Duration, the Duration Type, and the Power Level. In the example below the robot will travel 360 degrees at 50% power. Select compile and you will be prompted to save you file.

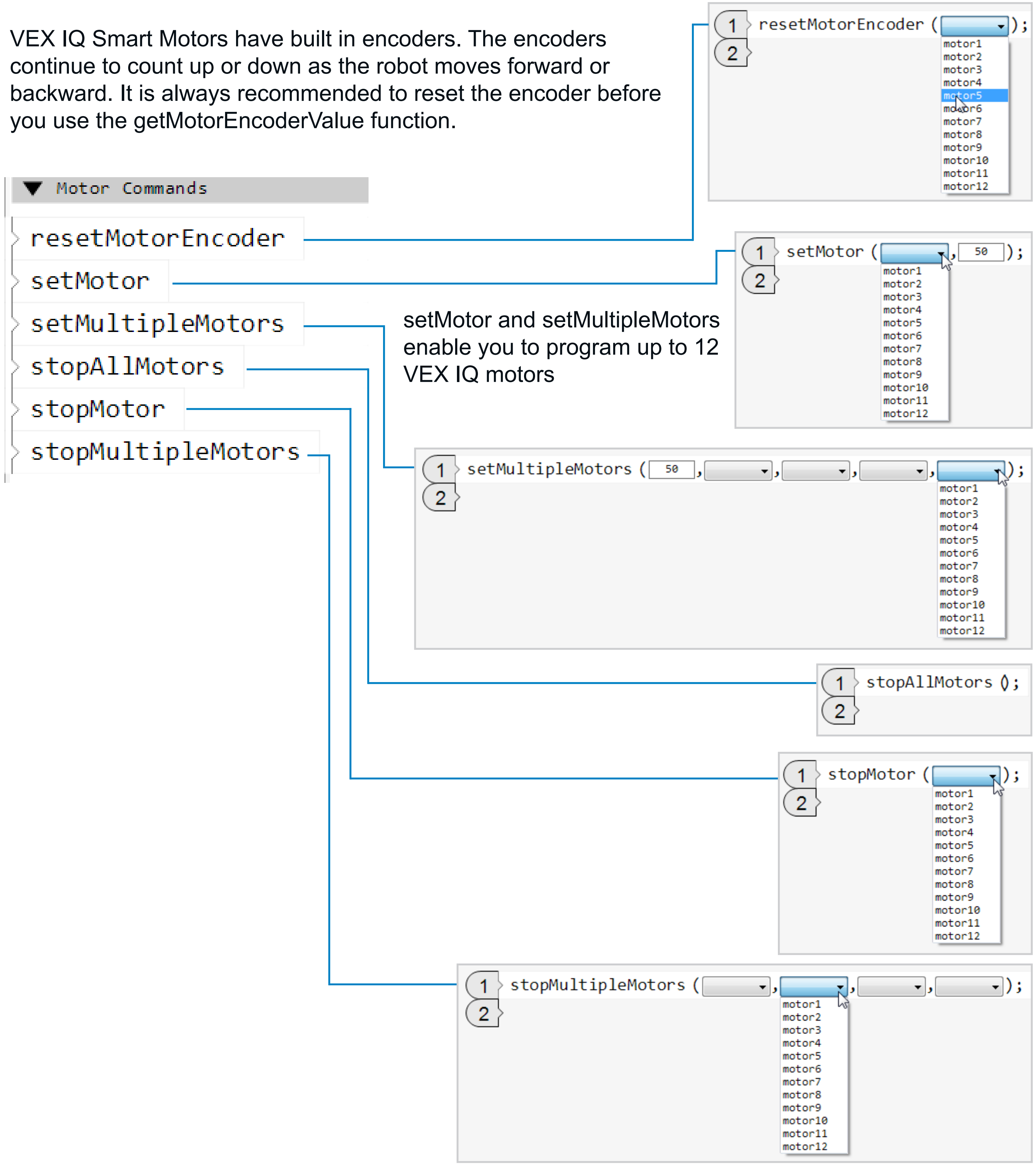


```
forward ( 360 , degrees , 50 );
```

Motor Commands

Motor commands give the programmer the flexibility that they need to program non-standard built robots.

VEX IQ Smart Motors have built in encoders. The encoders continue to count up or down as the robot moves forward or backward. It is always recommended to reset the encoder before you use the getMotorEncoderValue function.

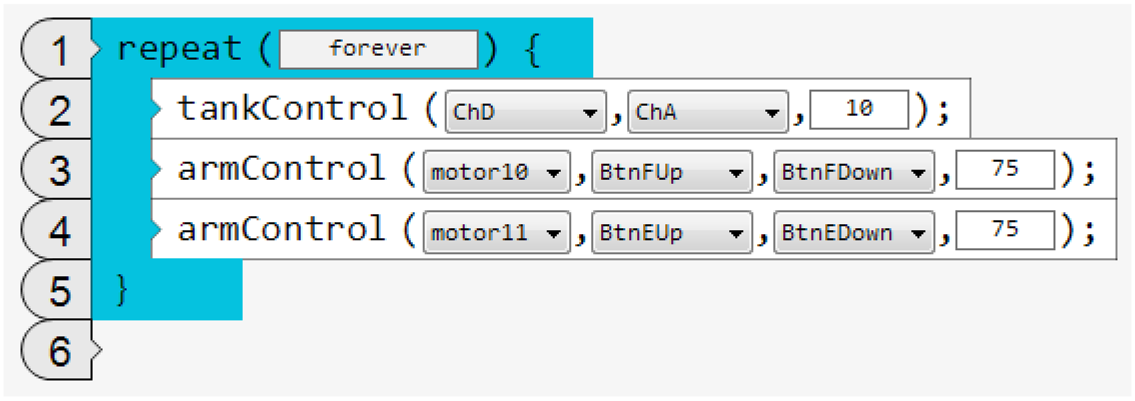


```
resetMotorEncoder ( );
setMotor ( , 50 );
setMultipleMotors ( 50 , , , , );
stopAllMotors ();
stopMotor ( );
stopMultipleMotors ( , , , );
```

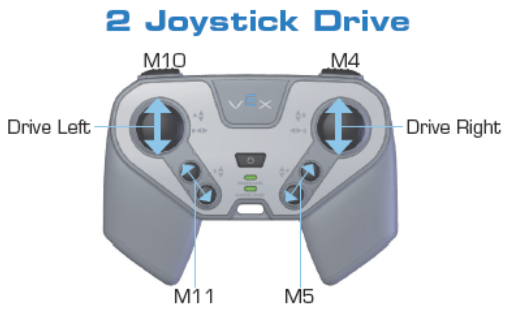
setMotor and setMultipleMotors enable you to program up to 12 VEX IQ motors

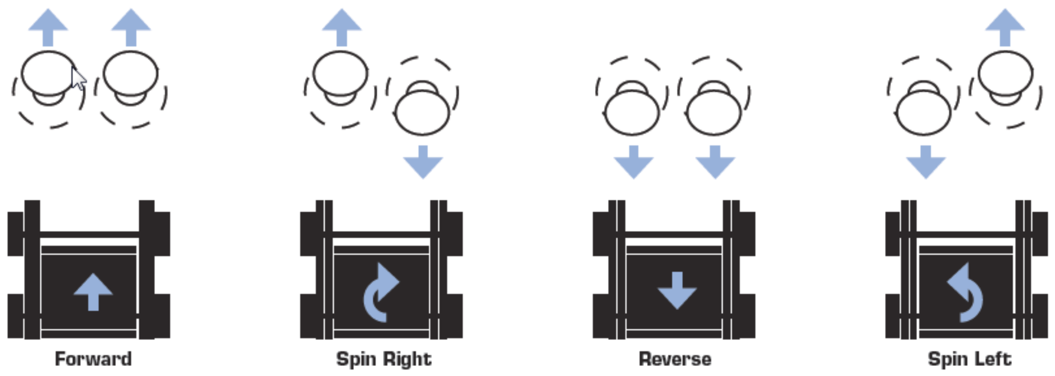
Remote Control

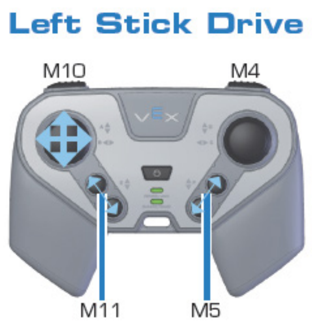
The most important thing for new programmers to remember is that when they use the Remote Control functions they must be placed within a loop to work. There are sample Remote Control programs included in the "sample program" folder.

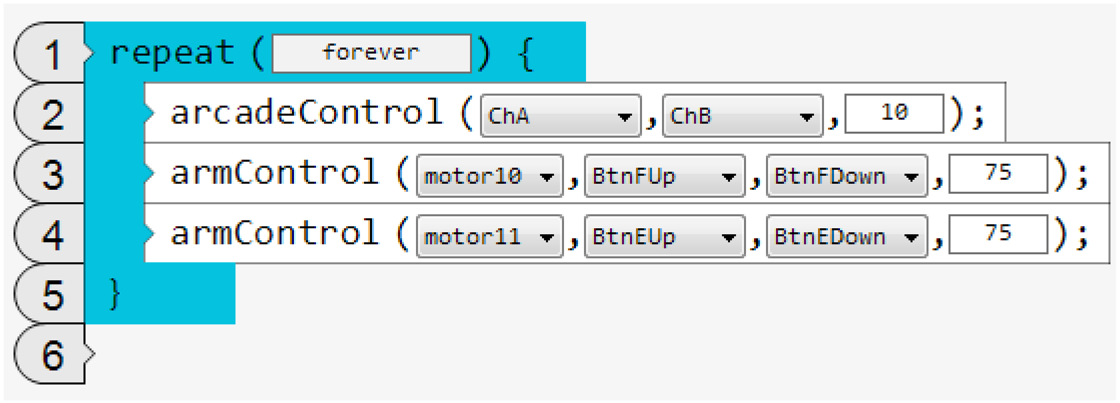


```
repeat (forever) {
    tankControl (ChD, ChA, 10);
    armControl (motor10, BtnFUp, BtnFDown, 75);
    armControl (motor11, BtnEUp, BtnEDown, 75);
}
```









```
repeat (forever) {
    arcadeControl (ChA, ChB, 10);
    armControl (motor10, BtnFUp, BtnFDown, 75);
    armControl (motor11, BtnEUp, BtnEDown, 75);
}
```

Can’t Compile and Download

1. Unplug the USB cable from both the computer and the robot, plug it back in, go to View->Select Communication Port->Click OK

2. Reboot the brain by doing a quick power cycle. If the brain is frozen, pop out the battery and put it back in.

3. If the error is telling you that the firmware is corrupted, follow step 1 and then re-download the firmware.