programming

table of contents:

table of contents, continued:

06/30/05

what is programming?

Technically, programming is the process of creating a sequence of instructions that tell a computational device, such as the Micro Controller on the Vex robot, how to perform a task. However, a programmer's real task is much broader and more involved than simply listing instructions for the robot to follow.

The programmer's true task is to analyze the problem at hand, and to identify the behaviors that the robot will need to perform in order to accomplish its task. The programmer must then break those behaviors down into simpler and simpler parts until they are at the level that the robot can understand directly, the level of a single easyC icon. The programmer will then organize those icons so that each simple behavior runs at the right time, and the desired overall behavior will emerge.

There is, of course, always more than one way to solve a problem, and it's up to you, the programmer, to determine what approach works best for your situation. By using your creative and analytical abilities to the fullest, you can build and program a robot to conquer any challenge!

setup overview

The Vex programming kit is a combination of software and hardware components that enable you to write programs for your robot and to then download them onto your Micro Controller.

You will program your robot on your computer with easyC®, the software included on the Programming CD that comes with the kit.

The Programming Kit includes both a USB-to-serial cable and a programming module. The USB-to-serial cable is used to convert your computer's USB port into a serial port. The serial end of the cable will then plug into the programming module, allowing you to download programs from your computer to the Micro Controller.

The following assembly pages will step through installing and testing the programming kit hardware and software.

For information and an overview of programming for Vex, see the *programming introduction* section.

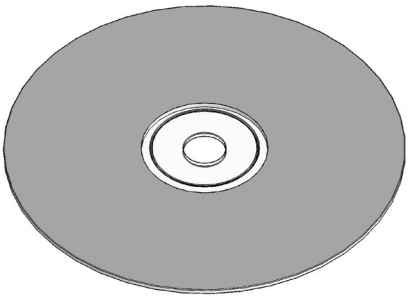
1 Collect and identify the parts from the list of materials below:

materials	qty
Vex Programming Kit CD	1
USB-to-serial adapter cable	1
robot interface cable	1
programming module	1

06/30/05

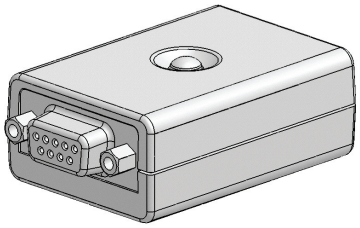
setup overview, continued

Vex Programming Kit CD x 1

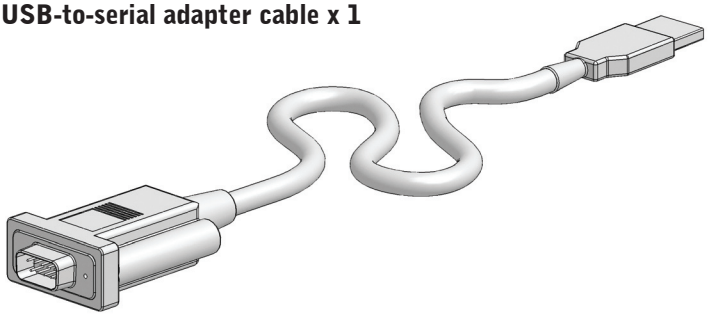


robot interface cable x 1

programming module x 1

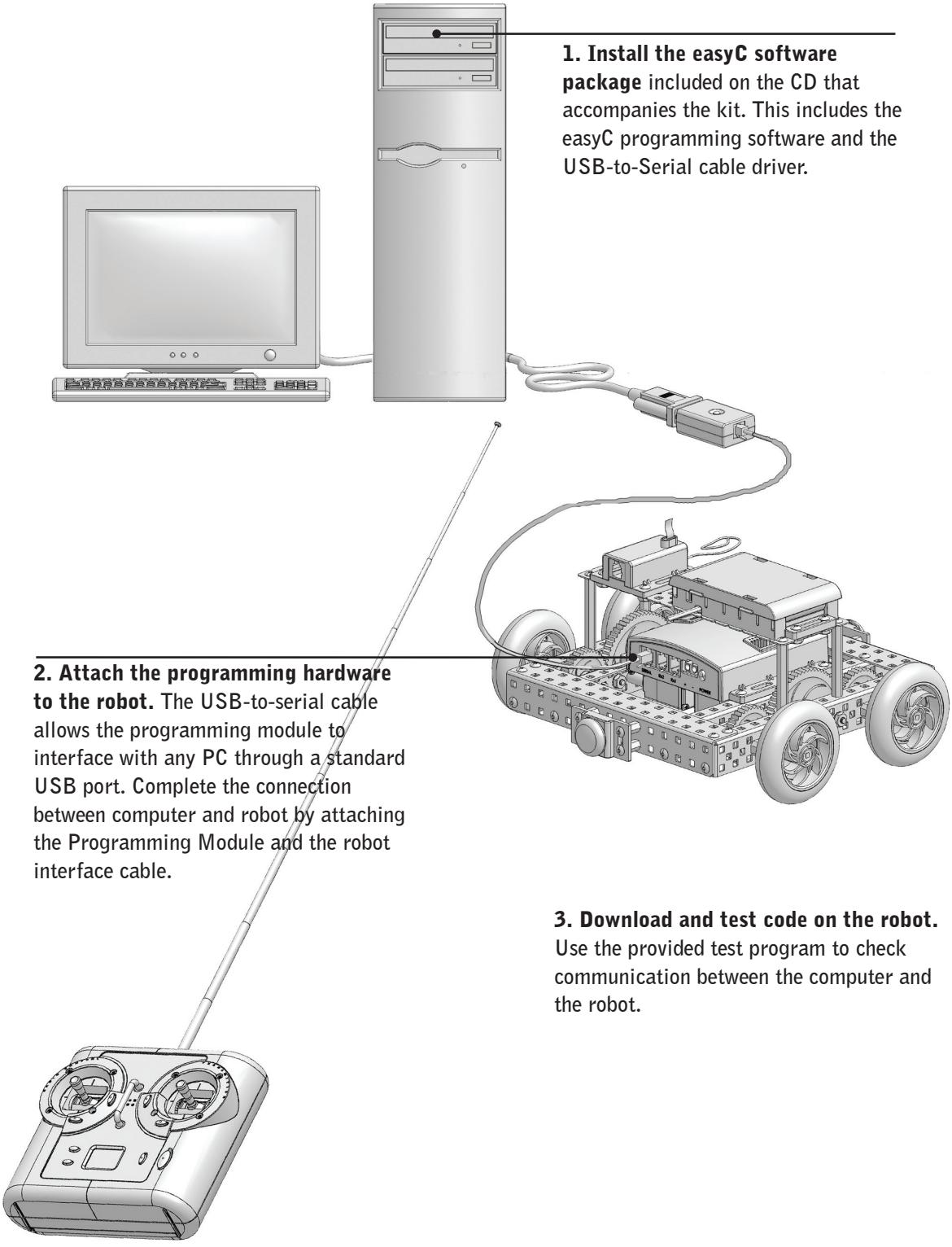


USB-to-serial adapter cable x 1



setup overview, continued

2 **Follow this recommended sequence to install the software and prepare your robot for programming:**



1. Install the easyC software package included on the CD that accompanies the kit. This includes the easyC programming software and the USB-to-Serial cable driver.

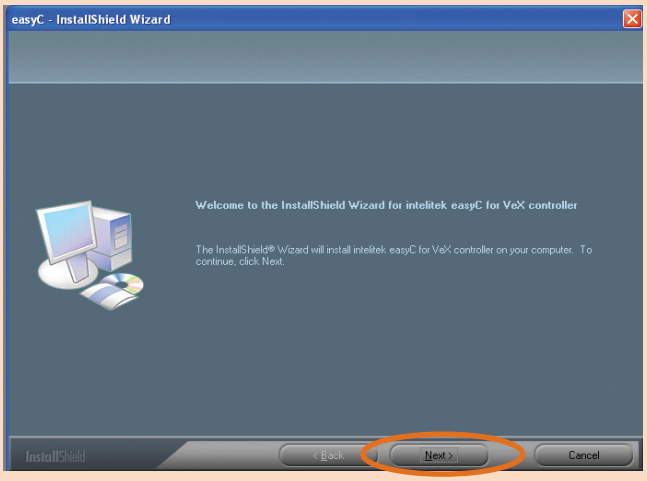
2. Attach the programming hardware to the robot. The USB-to-serial cable allows the programming module to interface with any PC through a standard USB port. Complete the connection between computer and robot by attaching the Programming Module and the robot interface cable.

3. Download and test code on the robot. Use the provided test program to check communication between the computer and the robot.

installing the easyC programming software

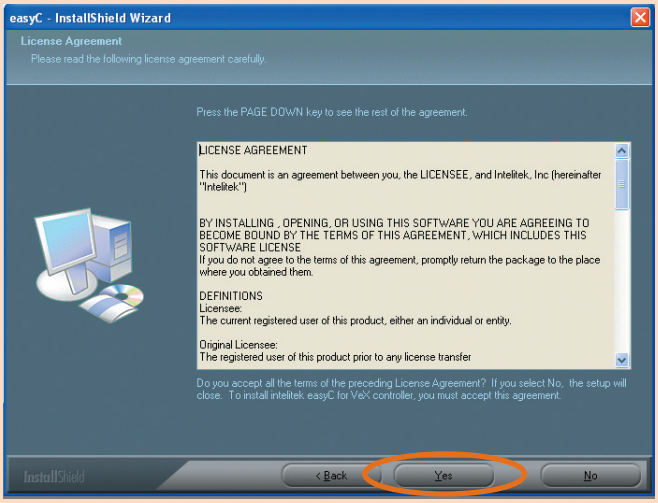
1. Insert the easyC programming CD into your computer's CD-ROM drive.

2. Your computer should automatically begin installing easyC. You should see the following screen. Click Next.



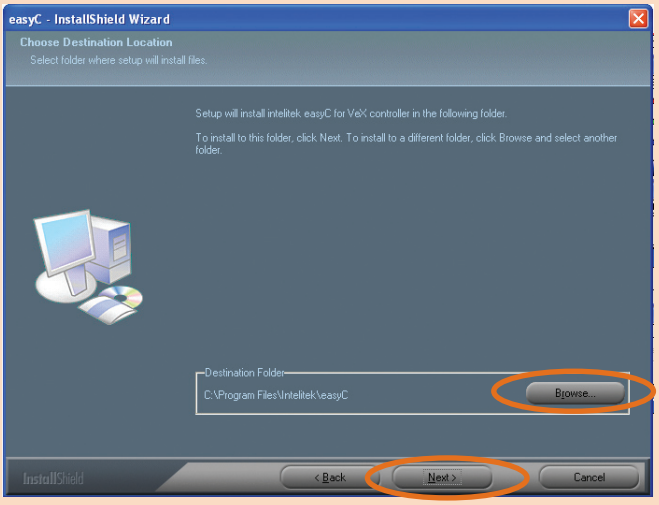
NOTE: If this screen does not appear, you may need to manually run the installer from the CD. You can do this by opening My Computer from your desktop or Start Menu and finding your CD-ROM drive. You can manually click the folder labeled "Install", then click easyC.exe.

3. At the License Agreement screen, read the easyC licensing. agreement. If you agree to the listed terms of use, click "Yes."

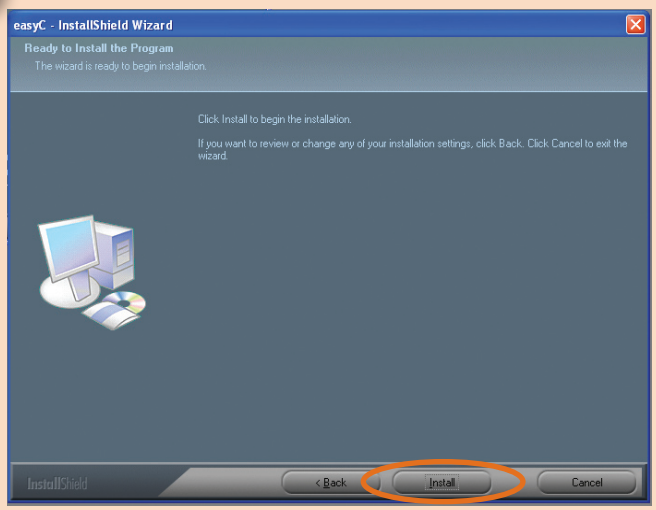


installing the easyC programming software, continued

4 If you need to install to a specific directory, or to a drive other than the C: drive, press the Browse button to select an install location, and then click Next to continue. Otherwise, just click Next to continue.



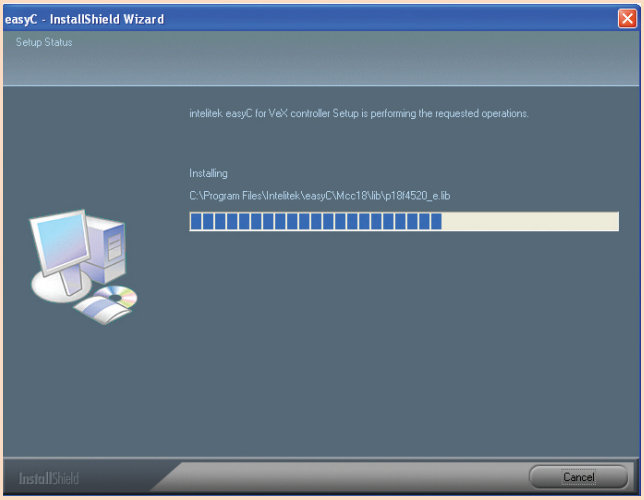
5 Click Install to begin the installation.



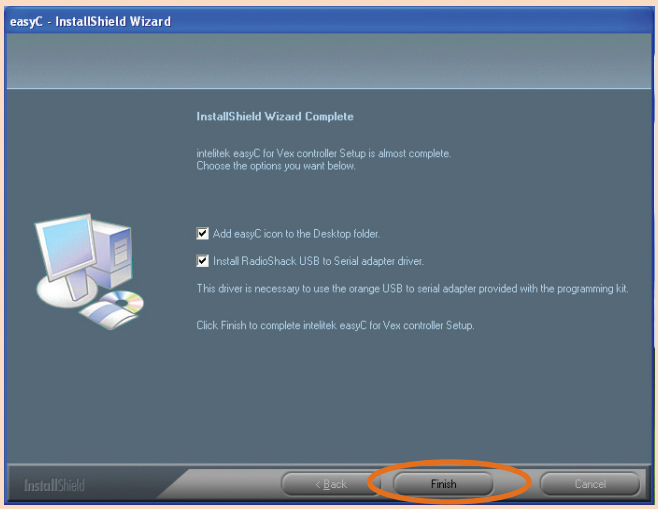
06/30/05

installing the easyC programming software, continued

6 easyC will be installed on your computer. This may take a few minutes.



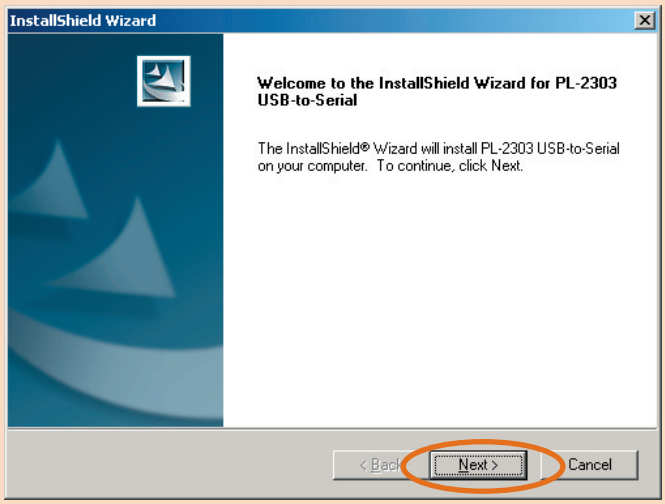
7 When the installation is done, you will see this screen. Check the box to install RadioShack USB to serial adapter driver. If you would like to add an easyC icon to the desktop, check that box as well. Click "Finish" to conclude the installation process of easyC.



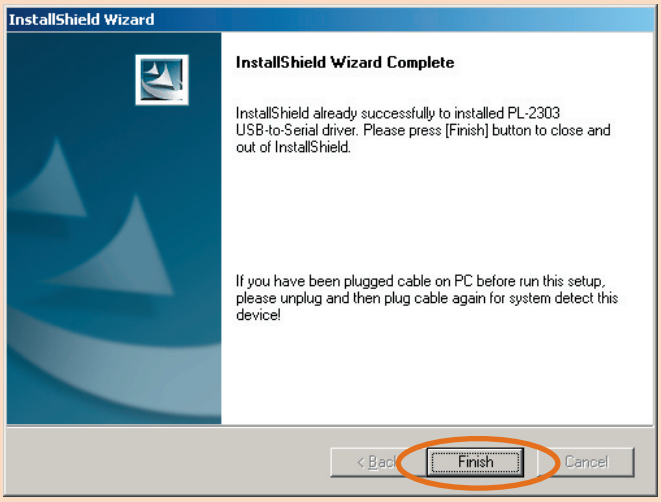
8 The easyC icon should now be accessible from either your desktop (if you checked the box for a desktop icon) or your Start Menu in the Programs or All Programs area, under "intelitek easyC for Vex controller".

installing the easyC programming software, continued

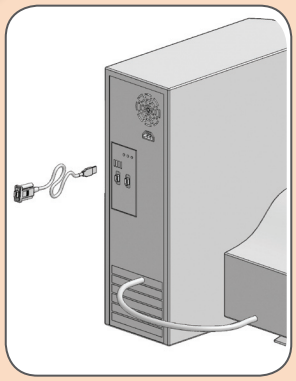
9 After installing the easyC software, the USB-to-serial cable software installation will start. Click Next.



10 The USB-to- serial cable software will be installed on your computer. Click Finish to complete your install.



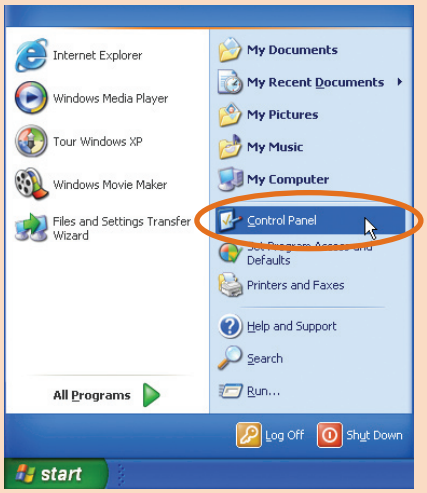
11 Plug the USB-to-Serial cable into your computer



06/30/05

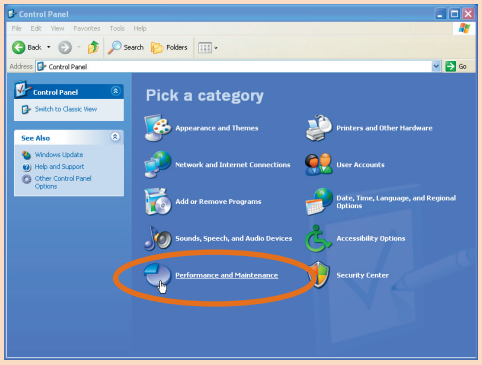
installing the easyC programming software, continued

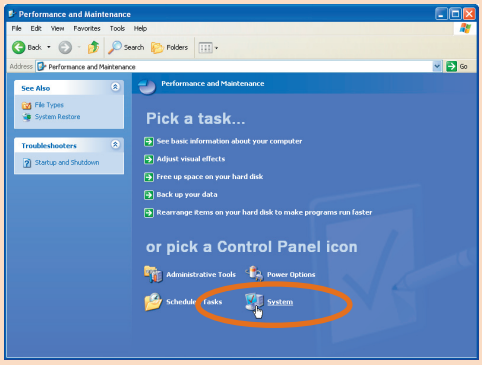
12 You will need to determine the COM port for your USB-to-Serial cable. To do this you must open the device manager.



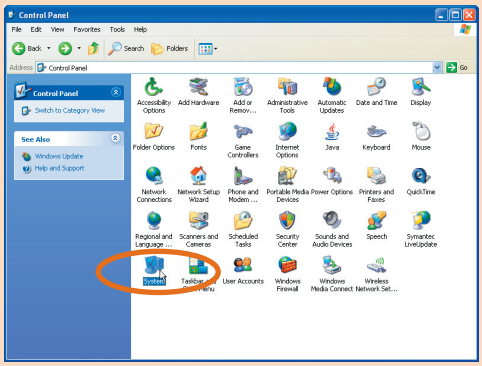
Single click on "Start" in the Windows Desktop.
Single click on "Control Panel" text.

The control panel may be in two different views. If it is in the "Category view" click "Performance and Maintenance" and then click "System".



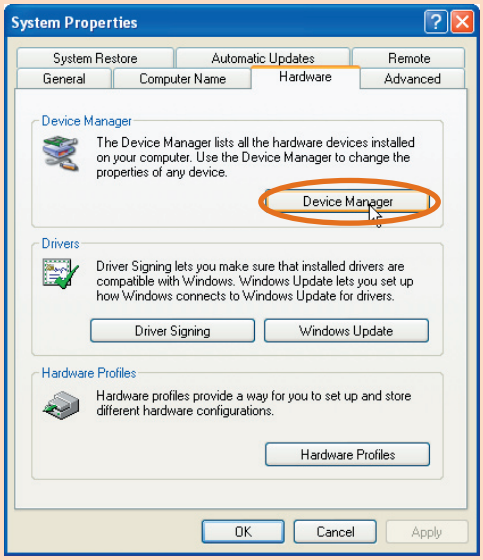


If the control panel is in "Classic View" simply double click the "System" icon.

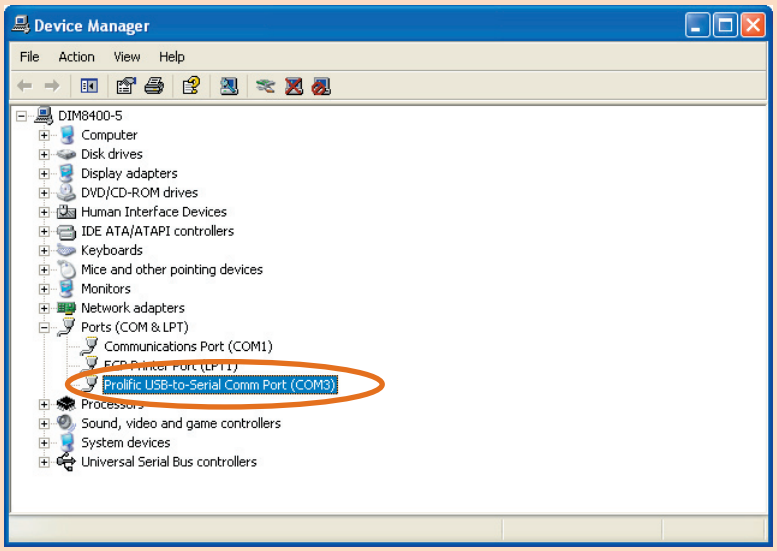


installing the easyC programming software, continued

Once you are into the "System Properties", single click on the "Hardware" tab. Single click on the "Device Manager" button.



Within the device manager do the following:
Double click on "Ports (COM & LPT)". Under Prolific USB-to-Serial Comm Port (COMx) make note of the number "x" as it will be needed to correctly configure your easyC software.



Close the "Device Manager" window.
Close the "System Properties" window.
Close the "Control Panel" window.

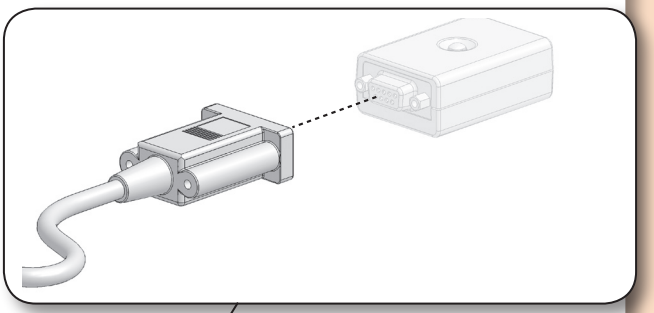
06/30/05

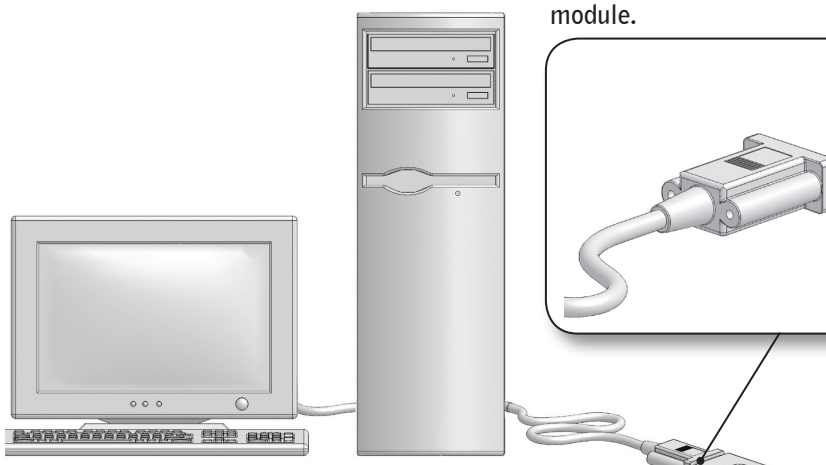
programming hardware setup

Connect your robot to your computer.

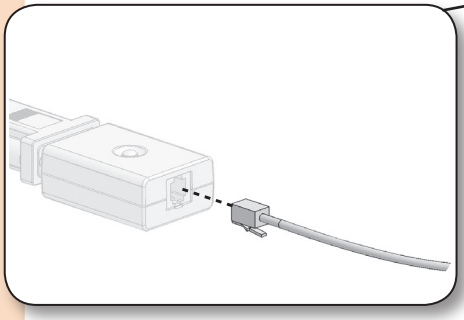
Now that the software is installed, you are almost ready to download the Vex test code to your robot. However, you still need to attach your Vex Micro Controller to your computer using the programming kit hardware. This setup will allow you to download code from your computer to your robot.

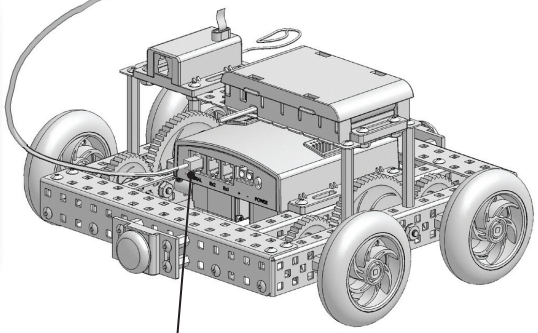
1. Attach the serial connector of the USB-to-Serial cable to the programming module.

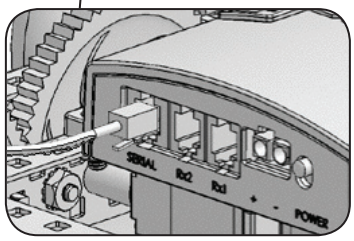




2. Attach the robot interface cable to the programming module.





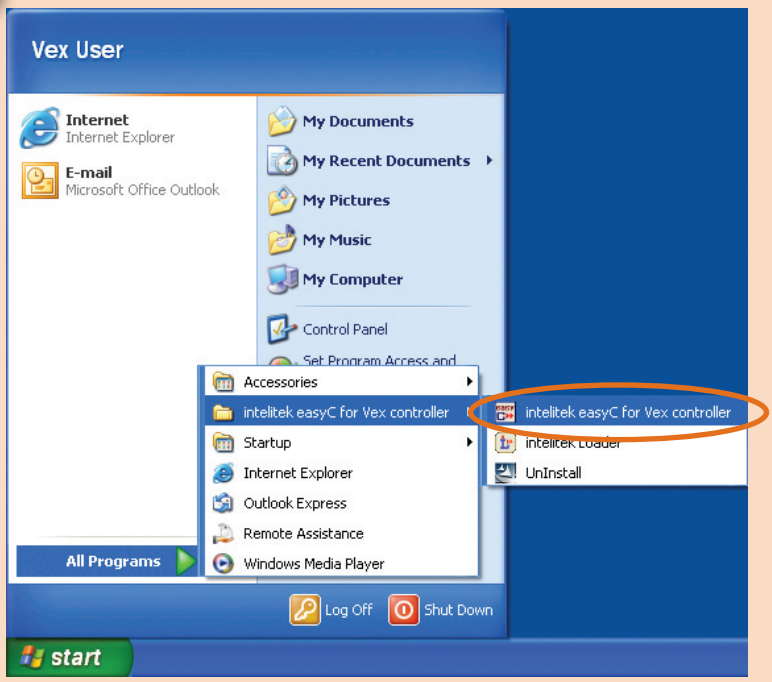


3. Connect the robot interface cable to the "Serial" port on the Vex Micro Controller.

download the test code

You are now ready to download the test code! This code will make sure your robot is set up properly and ready to be programmed.

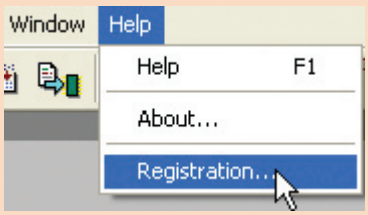
1 Open easyC.

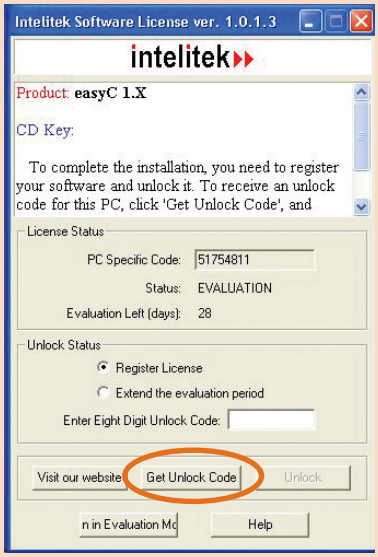


2 The evaluation mode screen will appear. Click the "Run in Evaluation Mode" button.

Note: You must register your software to continue using easyC after the 30 day evaluation mode is up. To register easyC, click the "Get Unlock Code" button and follow the instructions that appear.

To access the registration and licensing screen at a later time, click "Registration" in the "Help" menu.

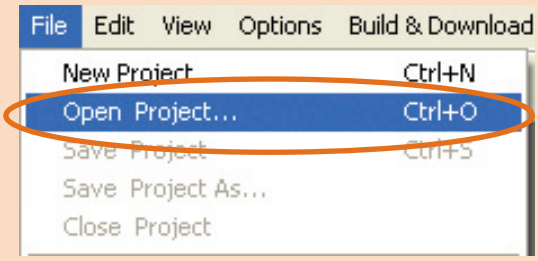




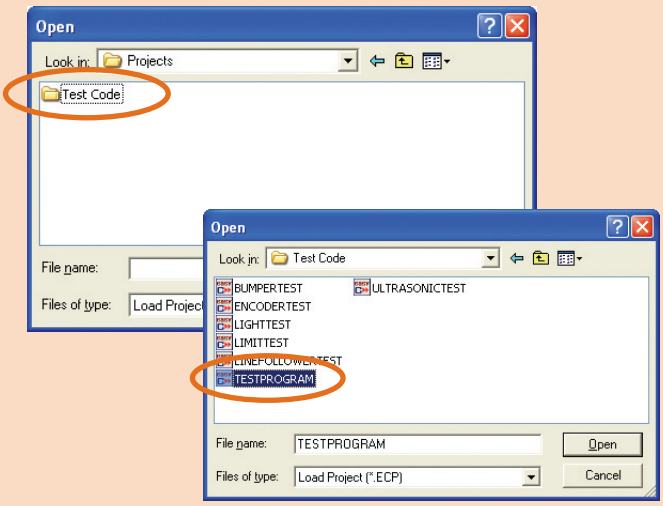
06/30/05

download the test code. continued

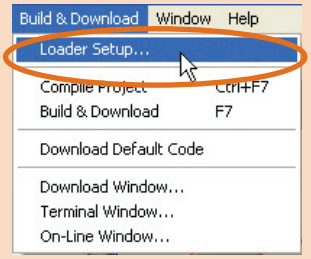
3 Click "File", then select "Open Project".



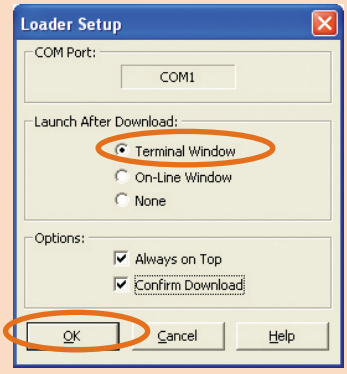
4 Open the folder called "Test Code". Find the file named "TESTPROGRAM.ECP". Double-click it.



5 Make sure the terminal window will appear by clicking the "Build and Download" menu and click the "Loader Setup" option.



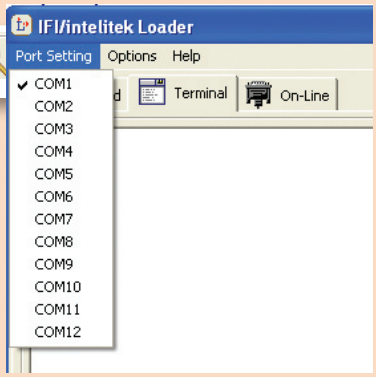
6 The "Loader Setup" configuration window will appear. In the launch after download section, select "Terminal Window". Click OK.



download the test code, continued

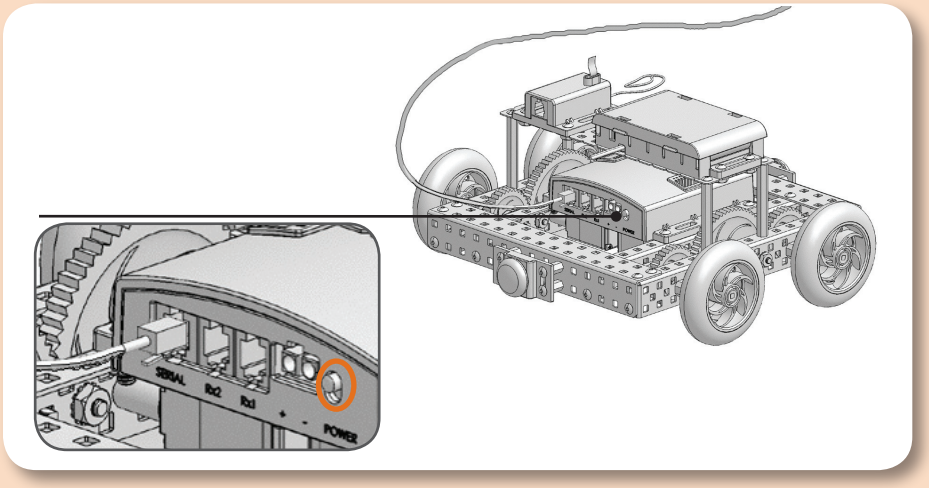
7 You must now set the COM port using the COM port information you found in the *installing the USB-to-serial adapter* section. To do this, open the Terminal window.

Now click on the terminal window icon in the toolbar.

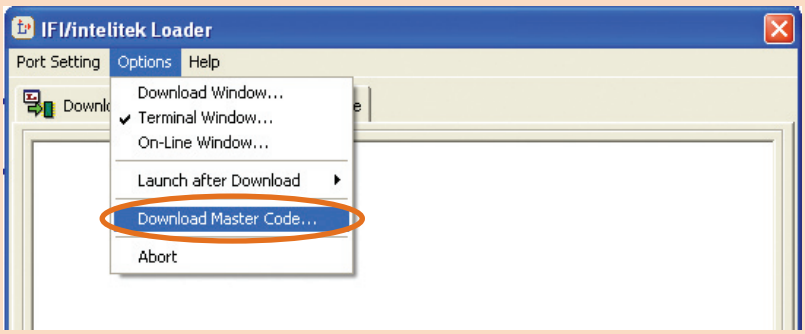


The terminal window will appear. Within the terminal window, click on the "Port Setting" menu. In this menu, click on the COM port you found in the *installing the easyC programming* section. If you were not able to locate your COM port, try different COM ports until downloading succeeds.

8 We recommend that you update the Master Code for best performance. The Master Code is responsible for the behind-the-scenes work inside the Vex Micro Controller. First, turn your robot on.



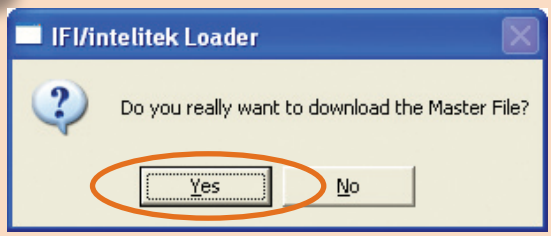
9 Since you are still in the terminal window, go to "Options" and click "Download Master Code...".



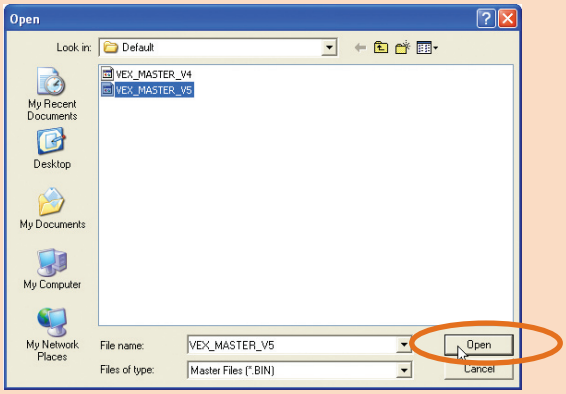
06/30/05

download the test code, continued

10 You will see a confirmation screen appear. Click "Yes".



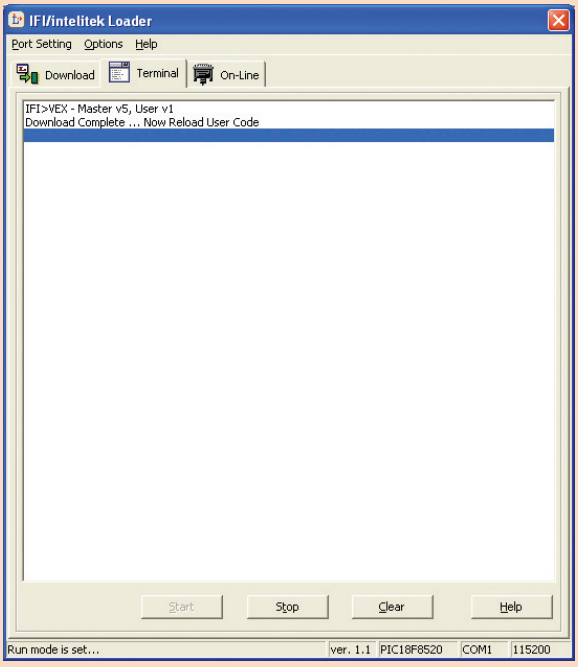
11 You will need to choose which version of the Master Code to use to update your Vex Micro Controller. Choose Master Code v5 and click "Open".



The computer will begin downloading code to your robot. This may take several minutes.

If you see an error message during compling or downloading, refer to *troubleshooting,* page 50 at the end of the programming guide

12 The terminal window will appear with the message "Download complete". You are now ready to download the test program to the robot!

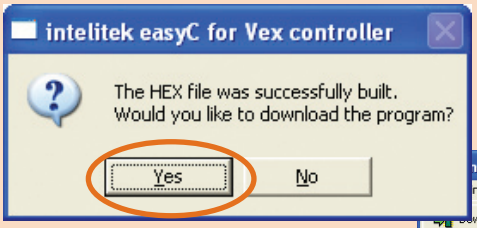


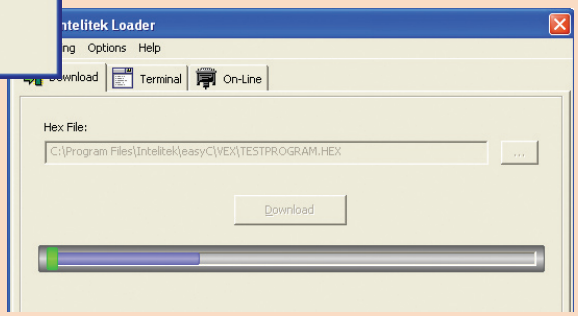
download the test code, continued

13 Click the "Build and Download" icon on the menu bar at the top of the screen.

14 You may see a confirmation screen appear. Click "Yes". Your computer will begin downloading code to your robot. This may take several minutes.

If you see an error message during compling or downloading, refer to *troubleshooting*, page 50 at the end of the programming guide

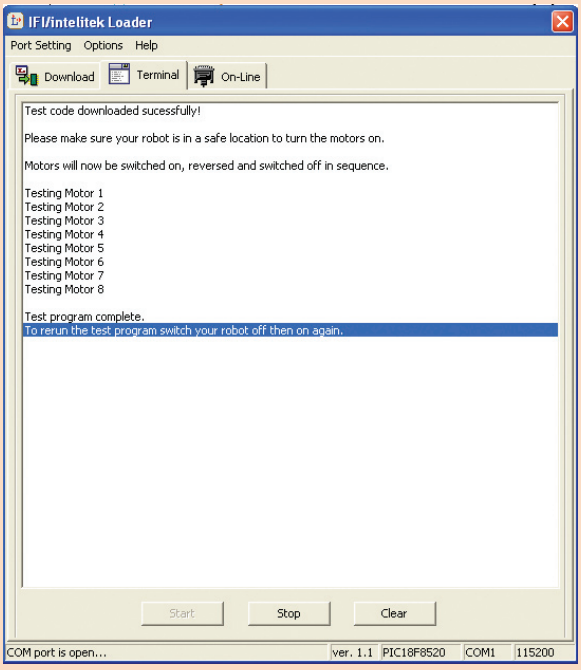




15 The program will begin to run immediately upon finishing the download. You should see the window below appear and text will begin to display. The program will test each of the motor ports (1 through 8) on your robot.

Any motor plugged into the Motor port bank will move briefly when its port number comes up.

If your robot is configured according to the "logic" section of your Vex Inventor's Guide, for example, then you will see motors 2 (right side) and 3 (left side) respond when those numbers come up, because you have motors plugged into those two ports.



programming options

To start you off and give you a safe "home base" to return to while you explore, a few key programs are already included with the easyC software.

Code provided for you:

Online Code (page 21):
This code allows you to control the motors and monitor the sensor values on your robot directly from your computer. The Online code is a valuable testing and troubleshooting tool.

Default Code (page 22):
This code is the default programming for the Vex robot (the program it came with fresh out of the box). This code is always available in case you ever need to return your robot's Micro Controller programming to its original state and get a fresh start.

The instructions on the following pages will show you how to load and use these included programs.

When you do begin developing your own robot code, you will find that your program's behavior will usually follow one of three main patterns:

Writing your own code:

Autonomous Code (page 27):
Autonomous code allows a robot to perform behaviors without input from the radio control transmitter.

Radio Control Code (page 45):
Radio control code allows you to configure the way in which the radio control transmitter controls the robot.

Mixed Autonomous and Radio Control Code (page 47):
Autonomous code can be integrated with radio control code to achieve even better robot performance for complex tasks.

The instructions and examples on these pages will guide you through some basic examples of each type of program behavior. These examples all use the same hardware configuration, found on page 13. Please take a moment to ensure your robot is set up properly.

All of the code written in the programming guide is compatible with the Squarebot design of the Vex robot. Although these programs may be applied to any Vex robot design, you may have to make small tweaks (like reversing the direction a motor spins) to get a robot other than Squarebot to perform the programs correctly.

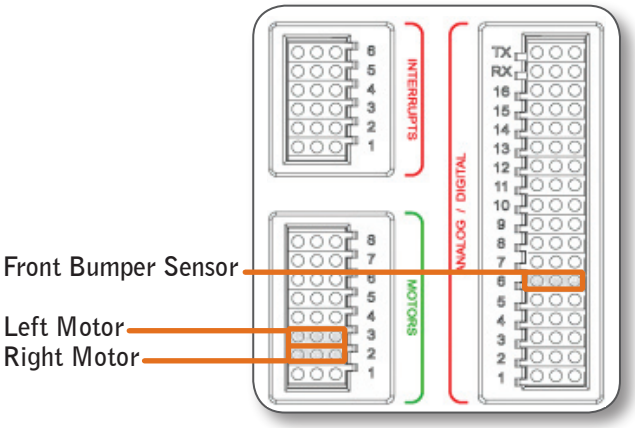
motor and sensor setup

Before you begin the programming examples in this section, please take a moment to ensure your robot's hardware is configured properly. All motor and sensor configurations are based on the Squarebot design presented in the assembly steps for the first six Inventor's Guide chapters.

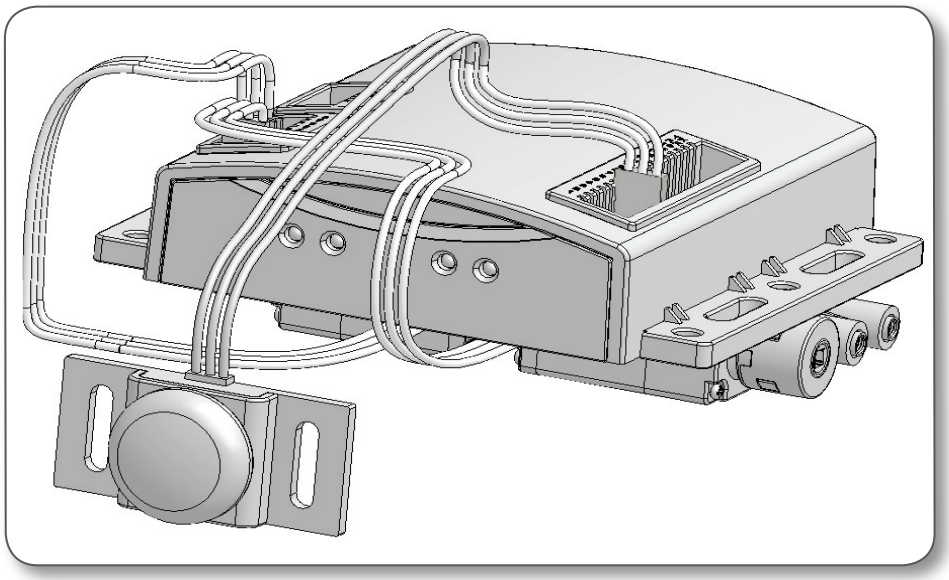
1. Make sure the left motor is plugged into Motor port 3 of your Vex Micro Controller.

2. Make sure the right motor is plugged into Motor port 2 of your Vex Micro Controller.

NOTE: For instructions on how to attach motors to your Vex Micro Controller, please refer to pages 6 and 7 of the "logic" section of your Inventor's Guide.



3. Check to see that the front-mounted bump sensor is plugged into Analog/Digital port 6.



4. The rear bump sensor is not used in the following examples. Please make sure the wire will not get caught in your robot's gears or wheels.

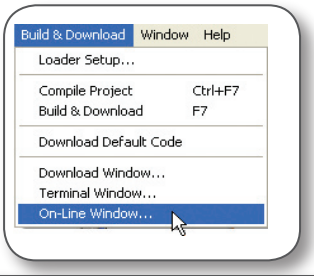
06/30/05

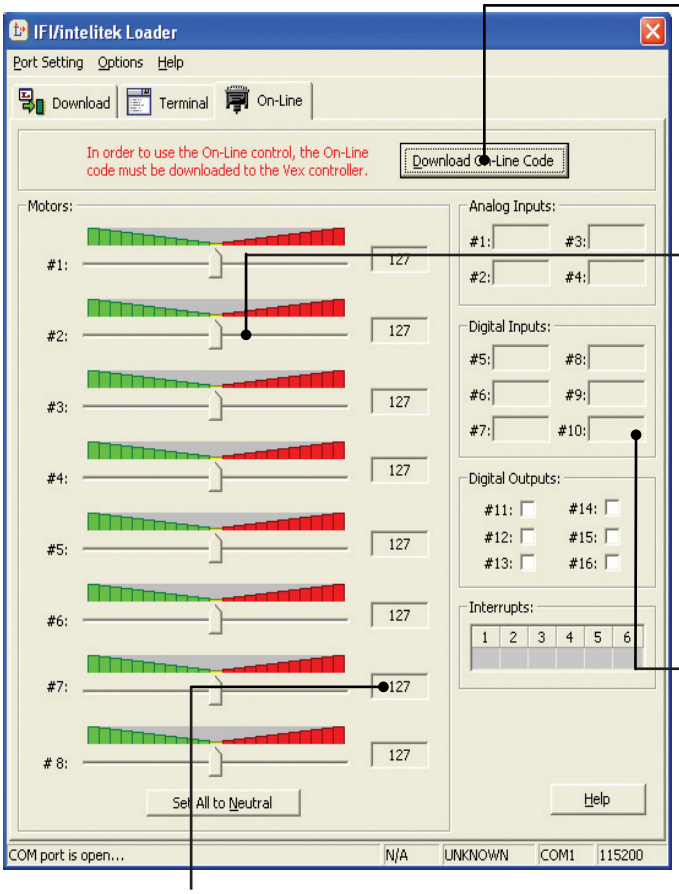
using online code to test your robot

This code lets you control the robot's motors and monitor its sensor inputs directly from your computer. This can be very helpful in troubleshooting problems with your robot or computer interface. To use the online code, follow the steps below.

1. Make sure your programming hardware is connected correctly (see *programming hardware setup*, page 13) and your robot is on.

2. Start easyC and select "Online Window..." from the "Build and Download" menu.





1. Click "Download Online Code". This will begin a download to your robot immediately.
NOTE: If your download does not complete successfully, visit *troubleshooting* on page 50.

2. You are now ready to begin using your computer to control your robot. Adjust the slider bars to make the robot's motors spin clockwise or counter-clockwise, or not spin at all.

If you have set up your robot according to the *motor and sensor setup*, then you will adjust slide bars 2 and 3.

3. As you slide in either direction, note that the number indicator changes. A setting of 0 means that the your robot's motor(s) will spin fastest counter clock-wise (CCW) and a setting of 255 will spin the motor(s) fastest clockwise (CW).

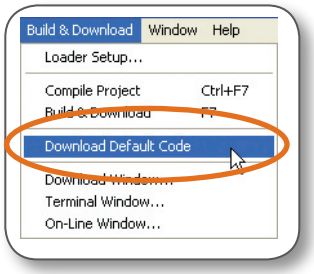
4. The Analog/Digital and Interrupt windows allow you to monitor the status of the sensors on each input port, or turn the output ports on or off. Try pressing the button on the Bumper Sensor and see what happens.

restoring default code

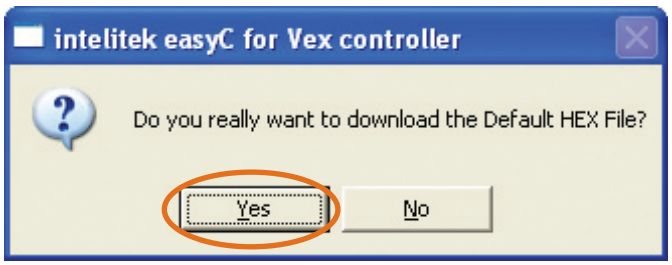
Should you ever need or want to restore the default functionality that your robot came with originally, follow the steps below.

1. Make sure your programming hardware is connected correctly (see *programming hardware setup*, page 13) and your robot is on.

2. Select "Download Default Code" from the "Build and Download" menu.



3. A confirmation screen will appear. Click "Yes." The default code will immediately begin to download to your robot.



4. Your robot will now function in its "out-of-box" state. Use the transmitter to operate the robot with its "out-of-the-box" functionality.

NOTE: For more information about the behavior of the default code, refer to the following sections of the Inventor's Guide:
- The Control chapter
- Appendix E - Control Configurations

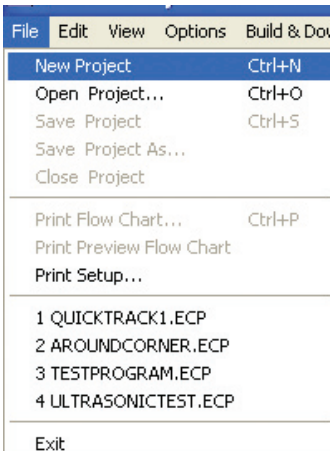
06/30/05

programming sequence

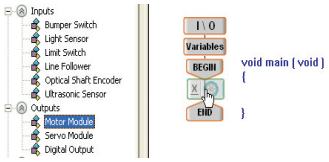
Now that you've had some practice downloading and using the included programs, it's time to learn to write some of your own! These are the three basic steps you will follow each time you create a new program for your robot.

1. Write Code

Write your program in easyC. Launch easyC and select "New Project" from the "File" menu.

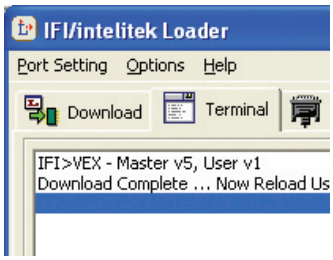


Drag function block icons into the program window to write or edit code.

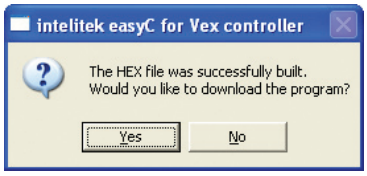


2. Compile & Download

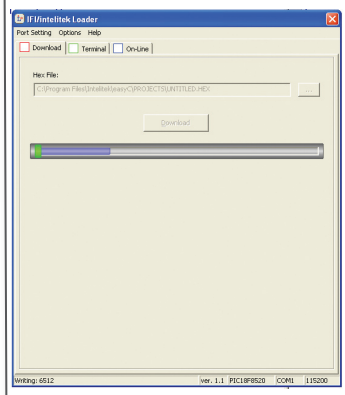
Once you have completed your code, make sure your programming hardware is set up properly and the robot is on. Then select the "Build & Download" option from the "Build & Download" menu.



A confirmation screen will appear. Click "Yes".



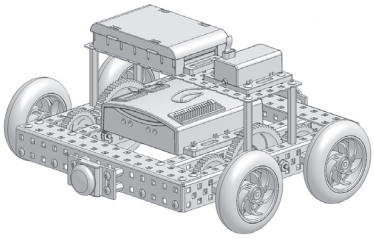
easyC will replace old code on the robot with new code you have written.



If you receive an error message during compile and download, please refer to the troubleshooting section.

3. Test

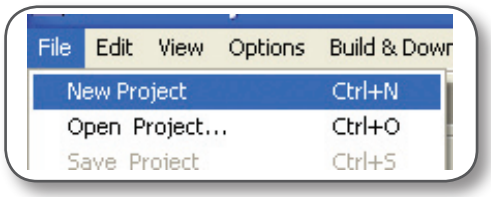
Run your code! Immediately after download, your robot will start executing the code. To re-run the code from the beginning, switch off your robot, and then switch it back on.



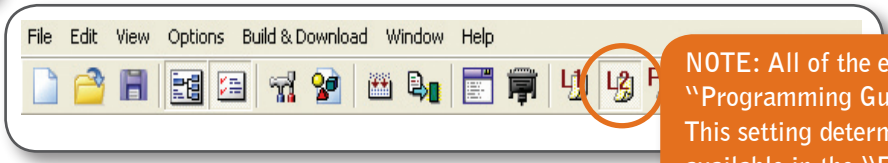
program one: writing your first program

Start simple. For your first program, you will learn how to turn a single motor on.

1 Launch easyC and select "New Project" from the "File" menu.



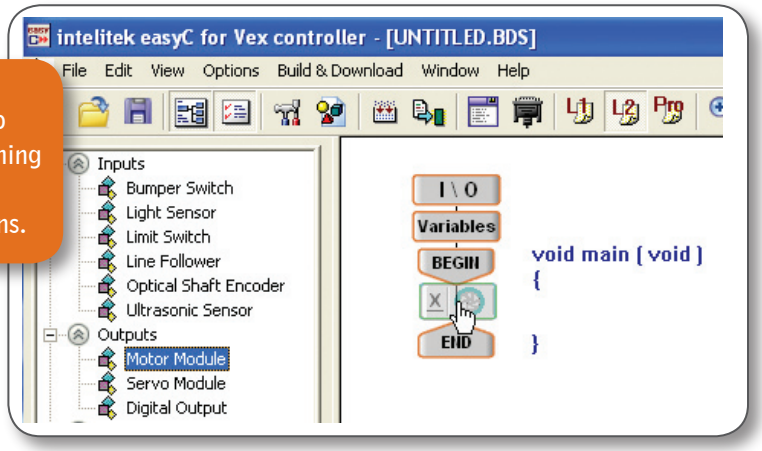
2 Select "L2" on the toolbar.



NOTE: All of the example problems in the "Programming Guide" use the "L2" setting. This setting determines what functions are available in the "Function Blocks" window.

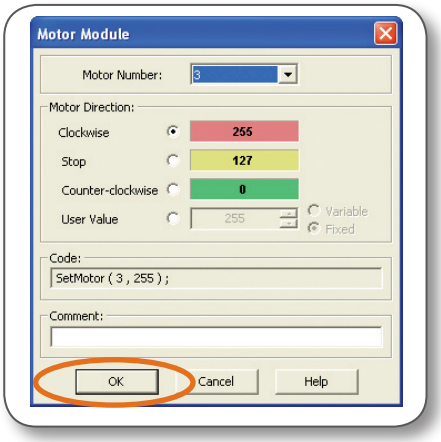
3 In the Function Blocks window (leftmost panel in the main window), under the "Outputs" heading, find the "Motor - Motor Module" block.

NOTE: Be sure to drag and drop your icons onto the line in the programming window between the "Begin" and "End" icons.



4 Left-click and drag the "Motor - Motor Module" block into the program window, between the Begin and End blocks. Release the block there.

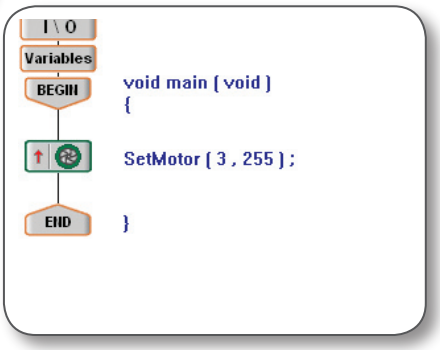
5 The motor module configuration window will appear. Make sure "Motor Number" is 3 and "Clockwise" is selected. Click "OK."



06/30/05

program one: writing your first program, continued

6 Your program should now look like this:



You are now ready to compile and download your first program!

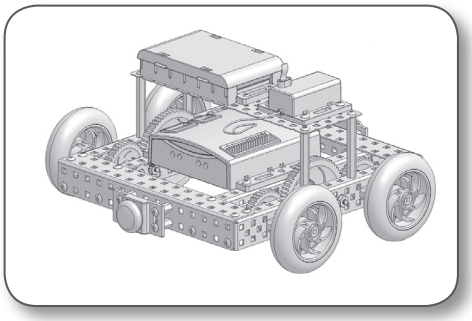
7 Make sure your hardware is set up properly and the robot is turned on. Select the "Build and Download" option in the "Build and Download" menu.

NOTE: If you receive any error messages, please refer to the *troubleshooting* section on page 50.

8 As soon as your program finishes downloading, the left motor will turn on, making your robot spin in place. To make it stop running, switch off the robot.

9 Now that your robot is off, you can unplug the wire from the serial port on the Vex Micro Controller and place your robot somewhere safe (for example, the floor). To run your program again, switch the robot back on.

10 Save your program as "intro program one" by following the steps on the next page.

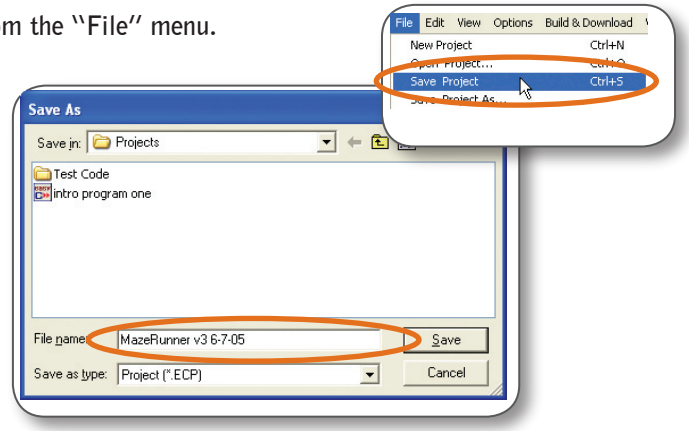


Congratulations, you have programmed your robot sucessfully!

tips for saving programs

In programming, it is very important to save your programs frequently. This will allow you to avoid losing work in case something goes wrong, and let you store programs at important milestones in their development, should you need to return to them later.

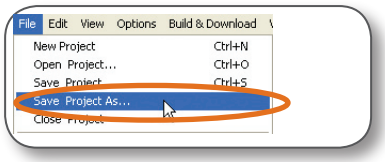
To save a project in easyC, select "Save Project" from the "File" menu.



The "Save Project As" window will appear. Enter the name you would like to save the program as. Be sure to name your programs such that you can easily tell which program is which as you write more programs. "MazeRunner v3 6-7-05" is a good name; "myprogram" is not.

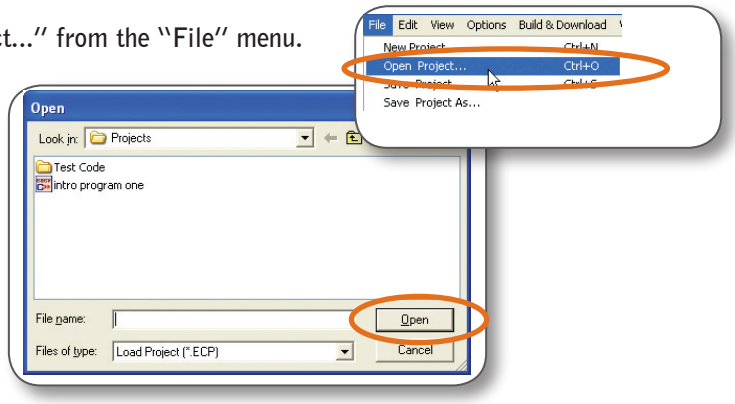
You may save programs to any directory on your computer, but we recommend using the projects directory to make it easier to find programs later.

As you write more complex programs you may find it useful to save multiple versions of the same code, so that you can revert back to an earlier version if your changes cause undesired behaviors. To save your project under a different name, select "Save Project As..." from the "File" menu.

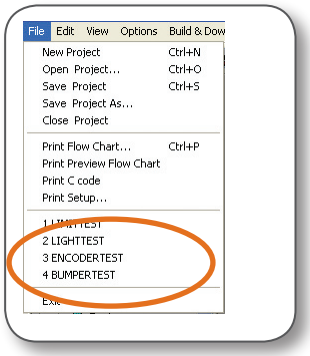


To Open saved programs Select "Open Project..." from the "File" menu.

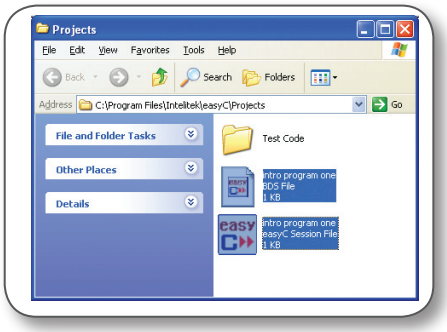
The "Load Project" window will appear. Find the program you would like to open, left-click on it, and click "Open".



The four most recent programs you have had open may also be opened directly in the "File" menu.



To move a program:
If you want to move a program, say from one computer to another, you must move both the .BCP and .ECP files.



06/30/05

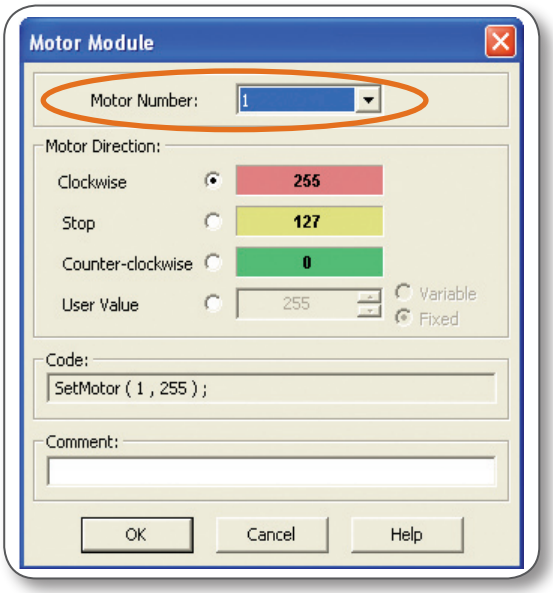
program two: using the motors

In the last section, you learned how to turn on one motor. Now, you will take things one step further by turning on both motors to make the robot move forward. In this section, you will write a program to move your robot forward for three seconds.

If you recall from the first program, a window with several motor control options appeared when you dropped the Motor Module icon into the program diagram. Here is a more thorough explanation of what all those options do:

Motor number - This specifies which motor will be controlled. This number refers to the motor port numbers (1-8) on the Vex Micro Controller.

With the hardware setup we're using (see page 11), the left motor is motor number 3, because the left motor is connected to motor port 3 on the Vex Micro Controller. The right motor is plugged into Port 2, so it would be motor number 2.

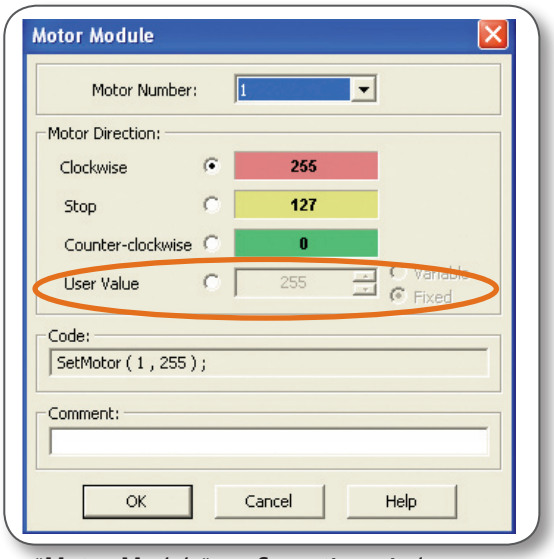


"Motor Module" configuration window

program two: using the motors, continued

Motor direction - This specifies which direction the selected motor will turn. Clockwise and counter-clockwise are oriented as if you were looking down into the hole where the motor shaft goes.

The User Value option allows you to specify an exact speed and direction value. See the *online code* section, page 21, for the meanings of these numbers.

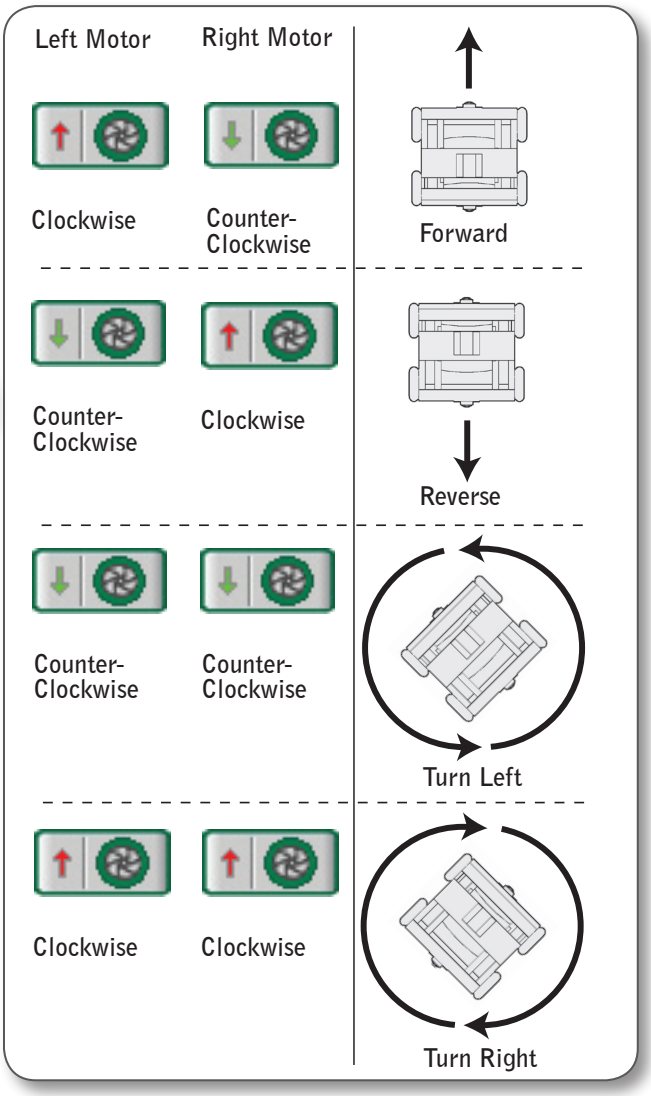


"Motor Module" configuration window

Squarebot has two motors, one controlling each side of the robot. This combination is enough to perform any of the basic movement functions. The chart at right describes the combination of motor movement needed to perform each maneuver.

Looking at the chart, you can see that to get the robot to move forward, you must set the left motor to "Clockwise" and the right motor to "Counter-clockwise".

To learn more about motors, refer to the Motion Subsystem Chapter of the Vex Inventor's Guide.

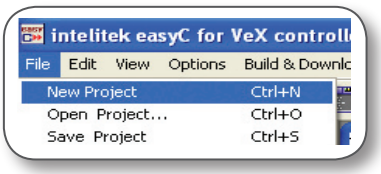


06/30/05

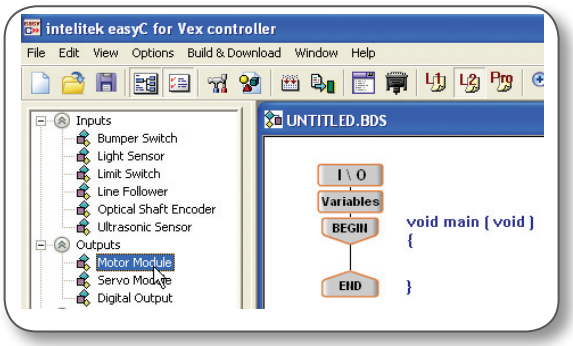
program two: using the motors, continued

Now it's time to write the program. The goal is to make the robot move forward for three seconds and then stop.

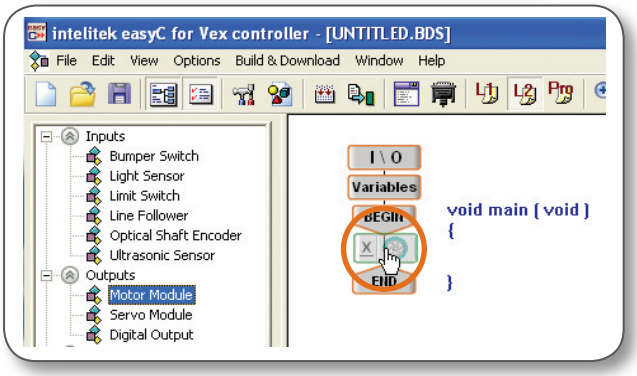
1. Launch easyC and open a "New Project" from the file menu.



2. In the "Function Blocks" window, under the "Outputs" heading, find the "Motor Module" block.

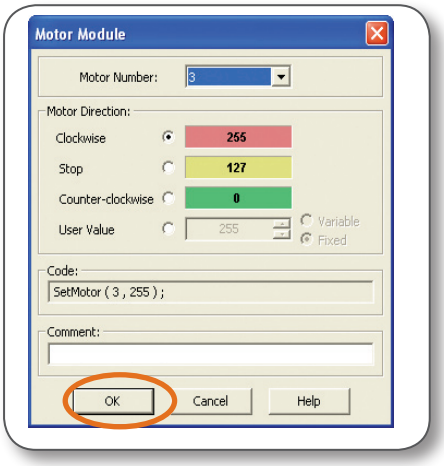


3. Left-click and drag a "Motor Module" block into the program window between the "Begin" and "End" icons. Release the icon.

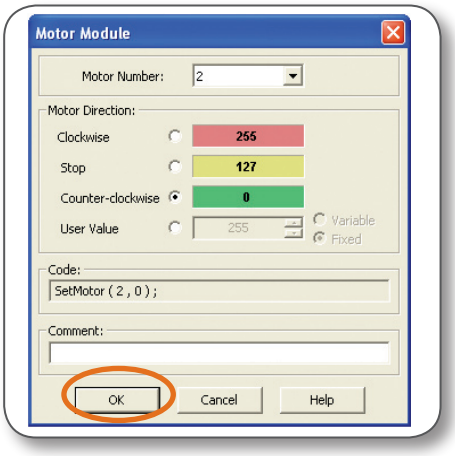


Reading from the chart on the previous page, forward movement requires the left motor to turn clockwise, and the right motor to turn counter-clockwise.

4. The "Motor Module" configuration window will appear. Make sure "Motor Number" is 3 and "Clockwise" is selected. Click "OK."



5. Repeat steps 2 and 3, dropping the second motor module icon right below the first. In the "Motor Module" configuration window, set "Motor Number" to 2 and select "Counter-clockwise". Click "OK".

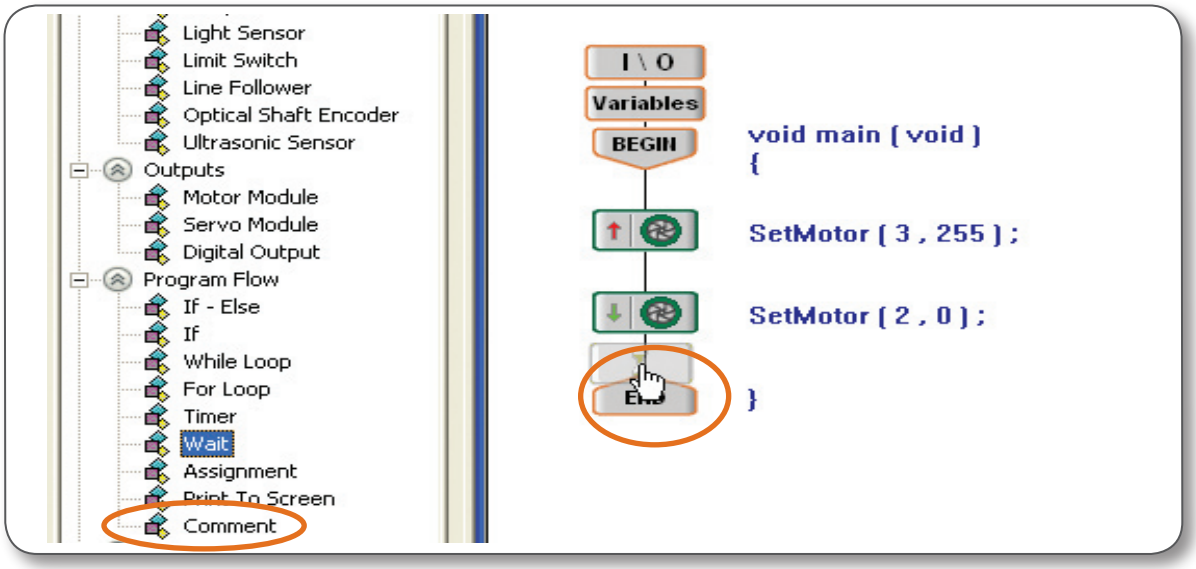


program two: using the motors, continued

Next, the robot needs to wait three seconds (during which time the motors will remain on).

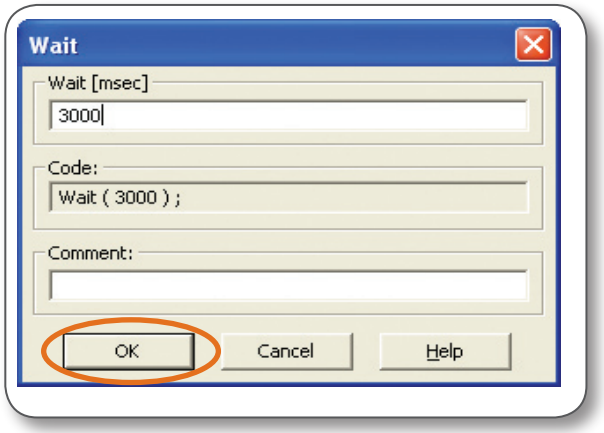
6 In the "Function Blocks" window under the "Program Flow" heading, find the "Wait" block.

7 Left-click and drag the "Wait" block into the program window. Drop the "Wait" icon on the line below the two motor commands so it will occur after the motors have been turned on.

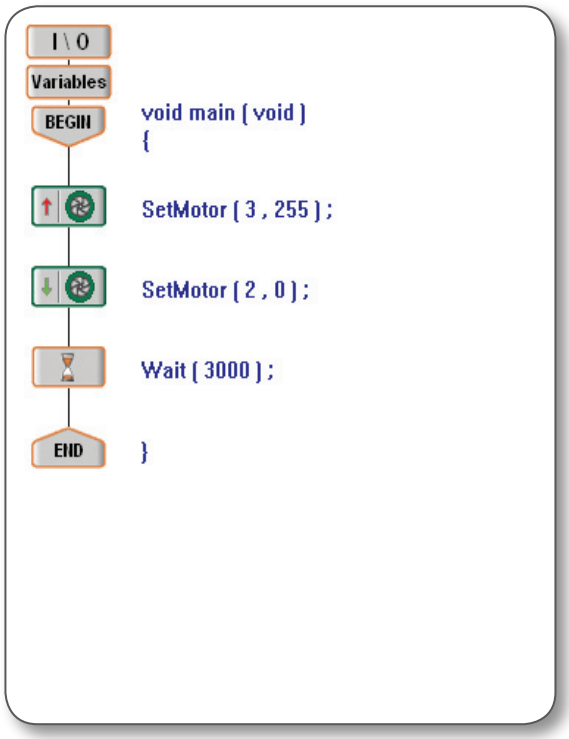


8 The "Wait" configuration window will appear. The "Wait" function is used to make the program wait a specified amount of time before proceeding to the next icon.

The default for the "Wait" is one second (1000 milliseconds). You want your robot to wait for three seconds, so enter 3000 into the "Wait[msec]" box, and click "OK".



9 Your program should now look like this :

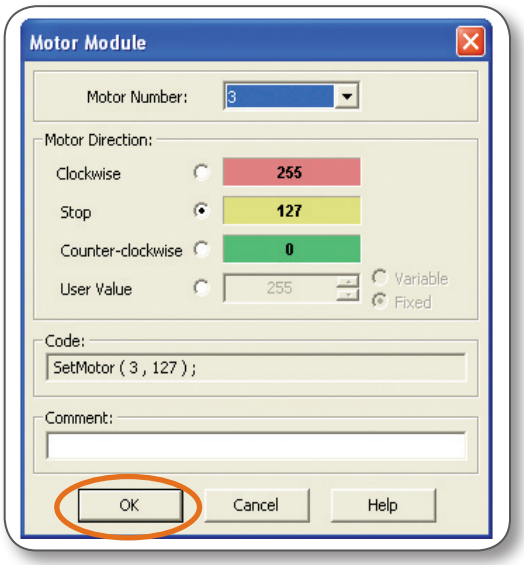


06/30/05

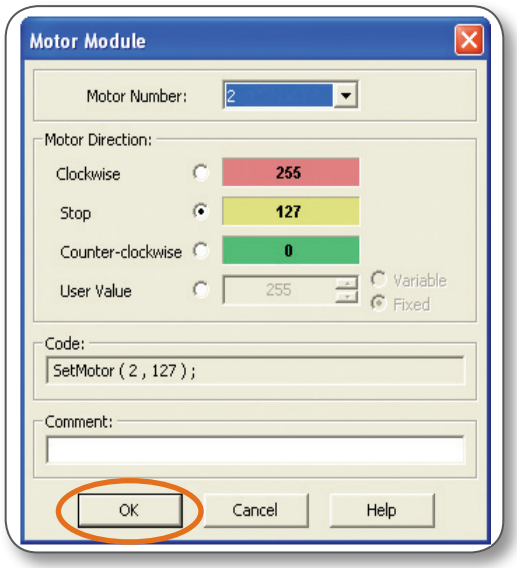
program two: using the motors, continued

So far, this program will turn on both motors and wait for 3 seconds. The final step is to stop both motors.

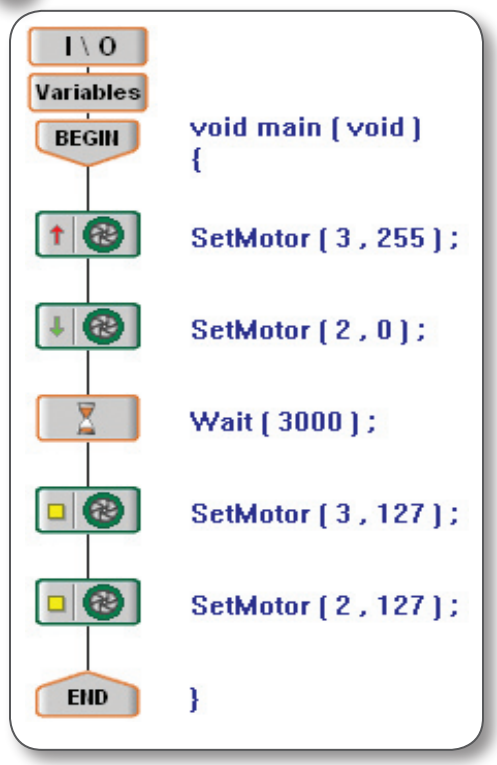
10 Repeat steps 2 and 3, dropping the new motor module block after the Wait block. When the "Motor Module" configuration window appears, set "Motor Number" to 3 and select "Stop". Click "OK".



11 Repeat steps 2 and 3, dropping the new motor module block just before the End block. When the "Motor Module" configuration window appears, set "Motor Number" to 2 and select "Stop". Click "OK".



12 Your final code should like like this:



13 Compile and Download your code.

Refer to the *Programming Sequence* section, page 23, for detailed steps on compiling and downloading.

14 When the program is finished downloading, your robot should move forward for three seconds, then stop.

15 Save your program as "intro program two". For a refresher on how to save your program, see *tips for saving programs* on page 26.

Congratulations, you have programmed your robot to perform a real behavior!

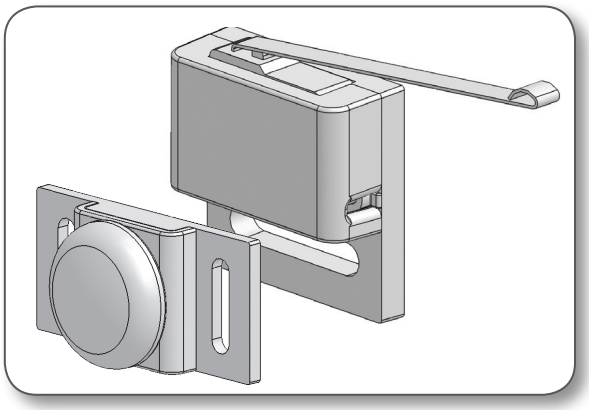
introduction to sensors

Sensors assist robots in seeing and feeling the physical world through which they travel. A sensor will tell a robot one very simple thing about its environment. The programs that you write will determine how the robot will react depending on the feedback from a sensor. Sensors can be placed in two different categories based on the type of signals they send or receive.

A brief review of sensors:

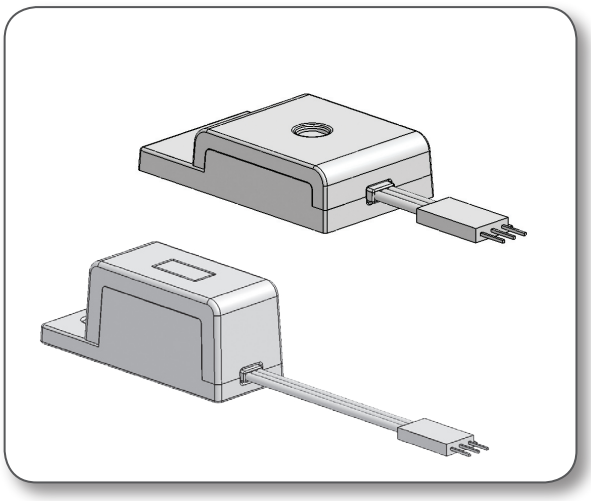
1. **Digital sensors** can distinguish between exactly two different "states". The exact meaning of the two states depends on the type of sensor being used. The robot will interpret information from a digital sensor as either 1 or 0.

   For example the Bumper Switch can read 1 (when it is not pressed) or 0 (when it is pressed). Similarly the Limit Switch can read 1 (when it is not pressed) or 0 (when it is pressed).



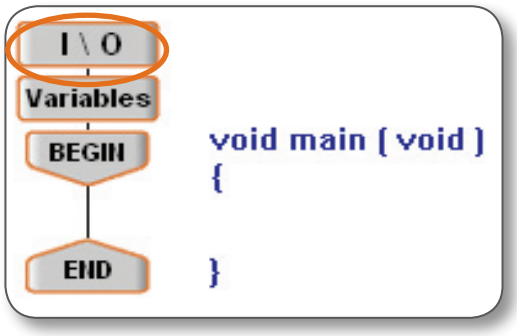
2. **Analog sensors** can provide a range of feedback. The feedback provided by an analog sensor ranges from zero to a pre-defined maximum limit for that particular type of sensor. The robot will interpert feedback from an analog sensor as a specific number between 0 to that maximium value.

   The Light Sensor and the Line Tracking Sensor, available separately as sensor accessory kits, are prime examples of analog sensors. They can read the relative "lightness" of a tabletop or other surface as a single numeric value. That value ranges from zero to one thousand.



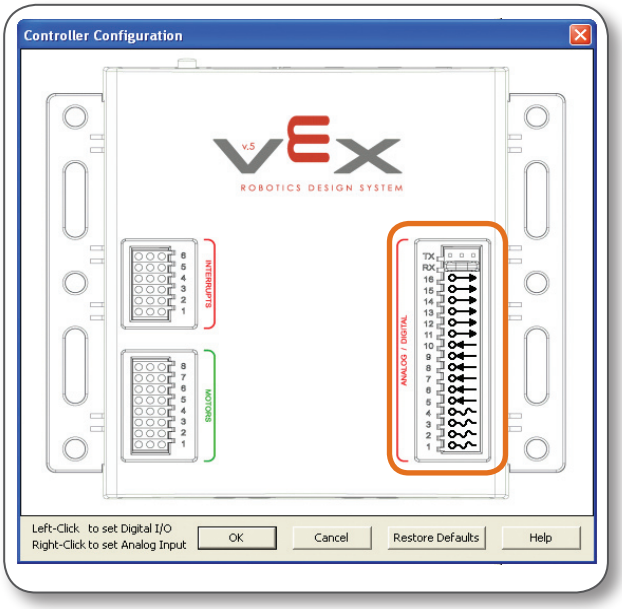
introduction to sensors, continued

The Analog/Digital Port Bank on the Vex Micro Controller consists of 16 ports that can be reconfigured to work as Analog Input, Digital Input, or Digital Output ports. The ports must be configured to match the type of sensor that is plugged into them (digital input for a digital sensor, etc.).



To configure the Analog/Digital ports, double click the Input/Output (I/O) icon located on your programming screen.

A window will appear revealing a blueprint of the Vex Micro Controller.



The section labeled "Analog / Digital" represents the sensor ports on your Vex Micro Controller. Left-clicking the ports switches the port to either digital input or digital output. Right-clicking these ports toggles them between analog and digital settings.

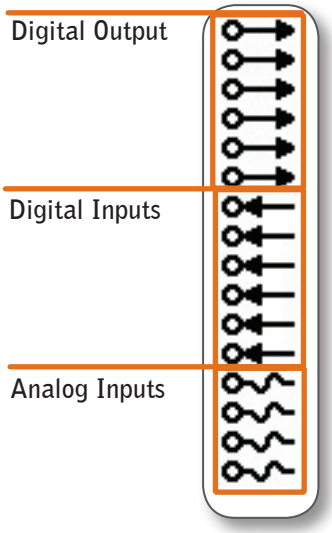
Note that the Analog ports must always be together on the bottom, so switching a port to Analog will automatically change all the ports below it to Analog as well.

Due to electrical load limitations on the Analog/Digital port bank, you cannot configure a port as an Analog Output.

Digital ouput ports are used for the Ultrasonic Sensor, and other 5V digital output devices, such as LEDs.

Digital input ports are used for Bump Sensors, Limit Switches, Jumpers, and other 5V digital sensors.

Analog inputs are used for the Light Sensor, the Line Tracking Sensor, and any other 5V analog sensor.

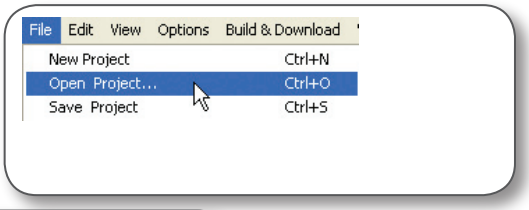


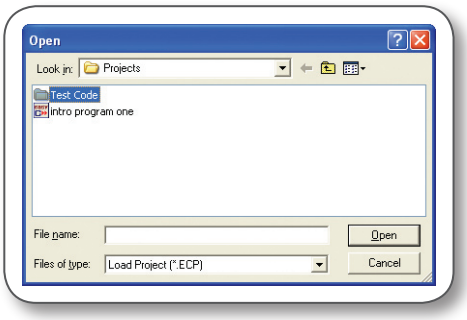
Refer to the *Sensors Subsystem Chapter* in the "Inventors Guide" or the *Sensor Accessory Pages* which come with Sensor Accessory Kits for more information.

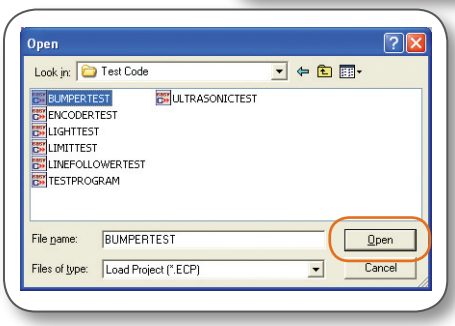
using a sensor

easyC includes a group of default sensor test files, which will allow you to test your sensors. First, you will compile and download a sensor test file to your robot to test it. Then, you will examine all the parts of this code and how they interact.

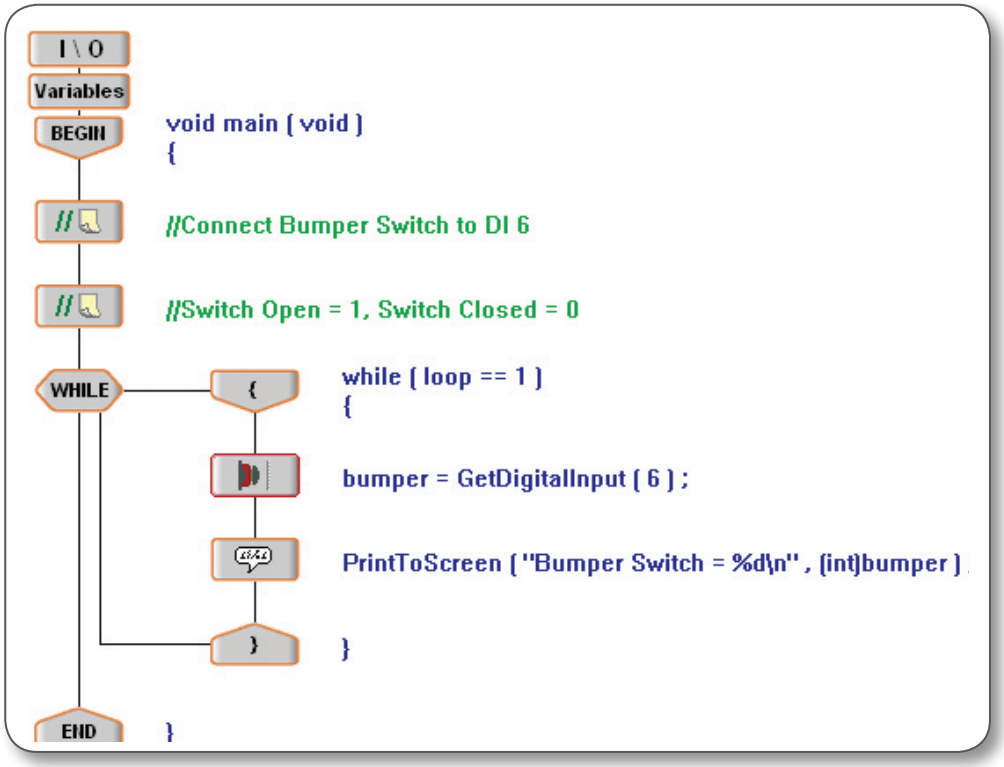
1 To open the test file for the bump sensor, click "Open Project..." in the "File" menu. Open the folder labeled Test Code, then click on the file called "BUMPERTEST". Then click "Open".







2 BUMPERTEST should open and your program window should look like the following:

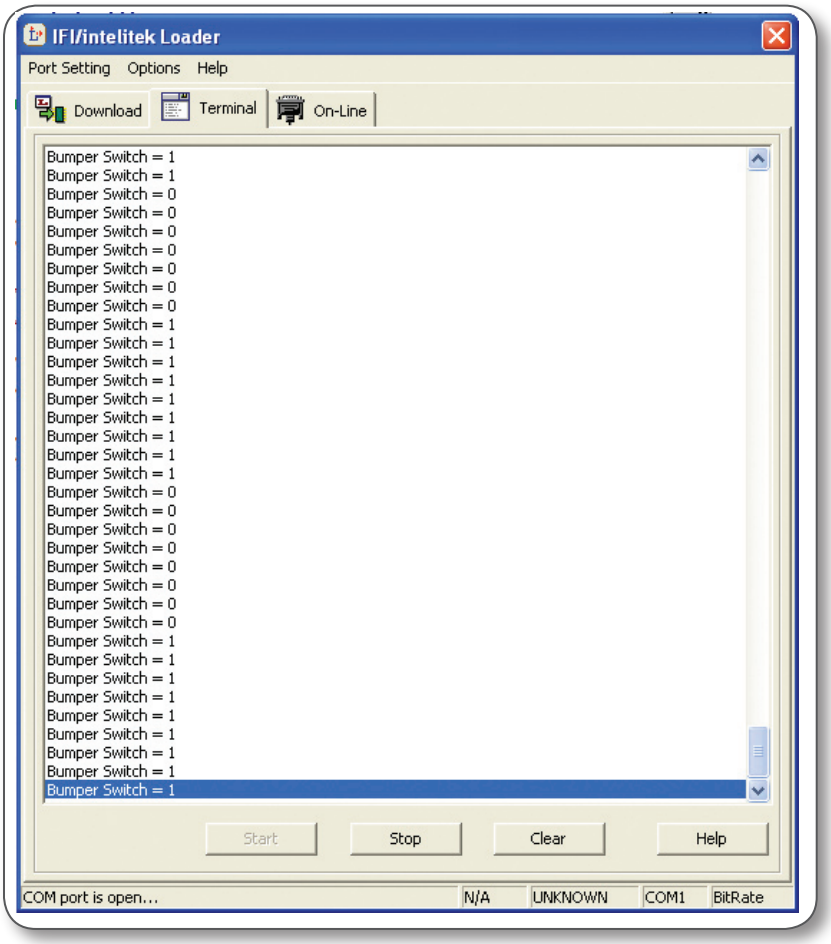


using a sensor, continued

3. Make sure your robot is set up according to the *motor and sensor guide* and your robot is on. Compile and download this code to the robot. (Refer to the *programming sequence* section earlier in this chapter for detailed steps on compiling and downloading.)

4. At this point, your robot should not be doing anything. The terminal window should open automatically. If it does not, click on the terminal window icon in the toolbar.

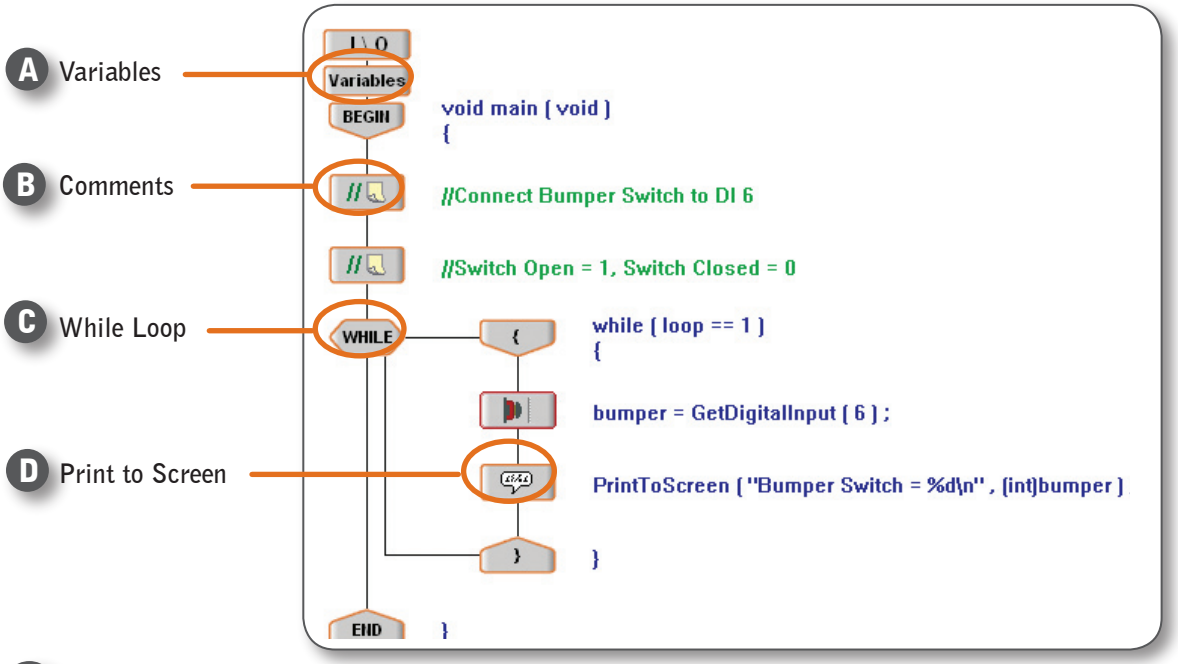
5. You should see the window below, with text scrolling across it. If you press and release the bump sensor, you should see the Bumper Switch output number change between 1 and 0. Press and release the bump sensor until you are familiar with the its operation.



The terminal window is the window used to display any information gathered from the robot, not only information from the bump sensor. If you have purchased any of the Sensor Accessory Kits, you will find that easyC also comes with test programs for all of these, and all the test programs will function very similarly.

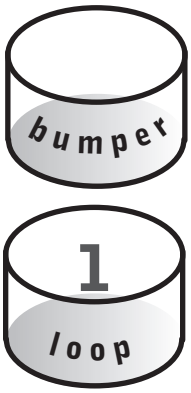
understanding bump sensor test code

Let's take a closer look at how the bump sensor test code works.

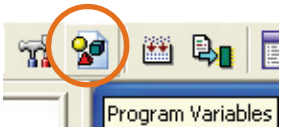


A

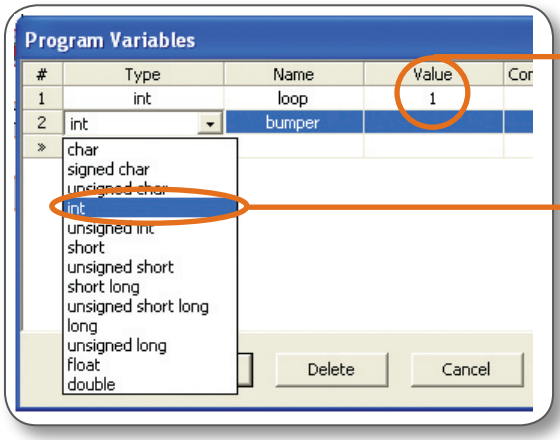
1. A variable is like a value-holding "container" with a name (label). Any time you need to store a value for later use in a program, you will need to use a variable to store that bit of information. You can then retrieve the value when it is needed.



2. To modify or create variables, open the Program Variables window by clicking the "Program Variables" button on the menu bar.



The screen below should appear:



3. If you enter a number into a variable's "Value" field, the variable will start with that number in it when the program begins. Here, the variable "loop" will have the number 1 in it when the program starts, whereas the variable "bumper" will have no value in it initially.

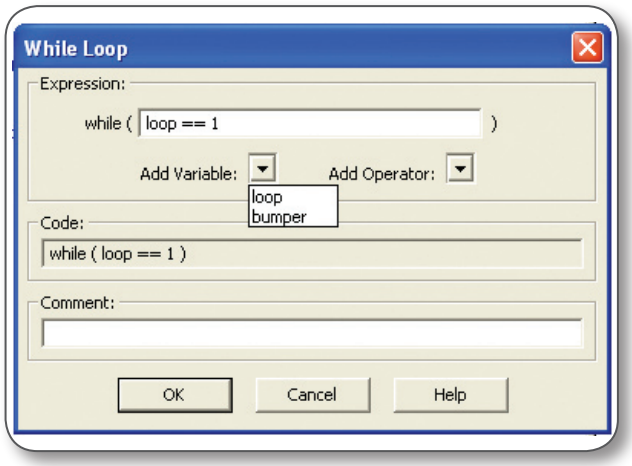
4. There are many types of variables, because there are different types of data to hold (a number versus a letter, for instance). For most basic Vex applications, you will use the "int" (integer) type for a variable, because sensor and timer readings both use this data type.

06/30/05

understanding bump sensor test code, continued

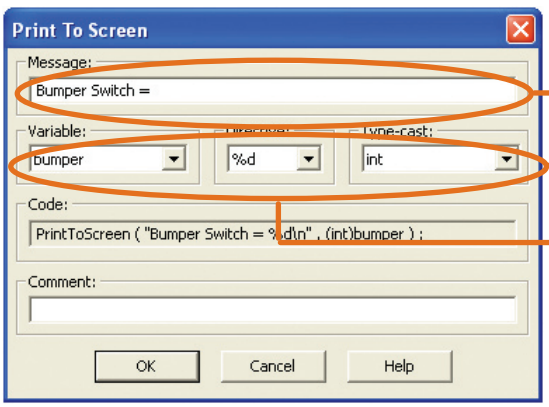
B **Comments** - Comments allow you to leave notes to yourself and others about your code. These comments do not affect how the program runs, but are very important in helping others to understand your code (and reminding yourself, if you revisit a program several weeks or months later).

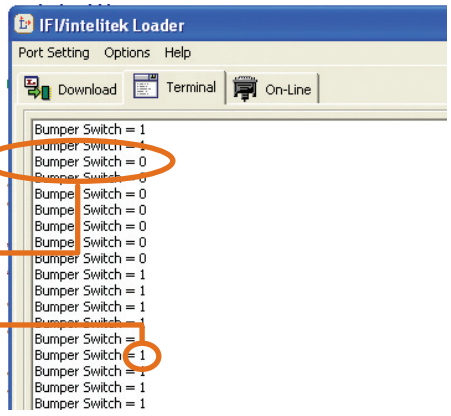
C **While Loops** - This is a block of code that will repeat itself as long as the condition inside the parentheses is true. The condition in the while loop in this program asks whether (loop == 1), i.e. whether the value of the variable "loop" is equal to 1. The code inside of this loop will run over and over as long as "loop" is equal to "1". As it turns out, the variable "loop" was set to a starting value of 1 in the variable window (see previous page). Since we never change the value of the "loop" variable, we can reason that it will always be equal to 1, and so this "while" statement will repeat its code forever (or at least as long as the robot is on.)



D **Print to Screen** - This is the code that allows you to see information in the terminal window. If you double click on the icon next to the printf() statement, you will see the "Print To Screen" configuration menu.

The "Message" field lets you specify text which you would like to display in the terminal window.



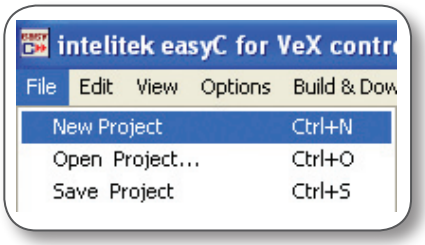


The optional "Variable" field lets you display the current value of a variable in the printed line (it will appear at the end, after the "Message" text). The "Variable" drop down menu lets you choose which variable to display. The "directive" menu tells the computer how to display the variable's value. %d tells the computer to use a decimal format, which works well for int-type variables like a bumper. The "typecast" menu specifies the type of the variable. This should usually match the type of variable you specified.

program three: using motors and sensors together

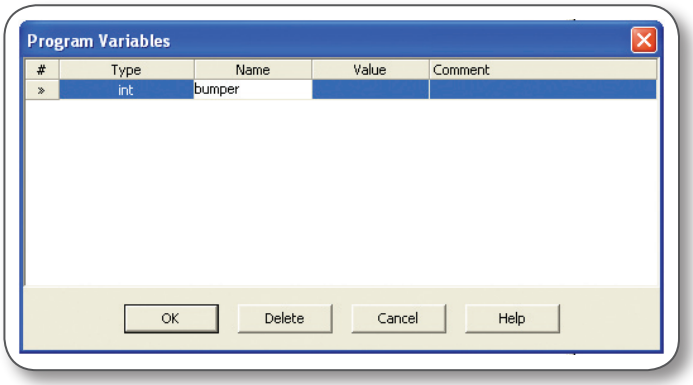
This program will make your robot move forward forever, unless the bump sensor is pushed in.

1. Start a new project in easyC by clicking "New Project" in the "File" menu.

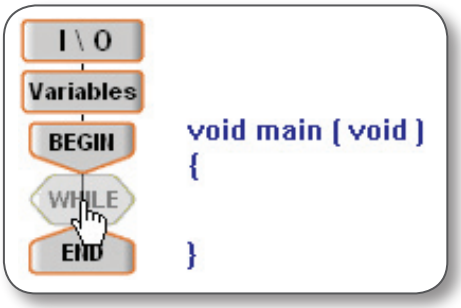


2. Double-click the Variables block in your program to open the "Variables" window. Create a new "int" type variable named "bumper", then click OK.

Refer to the *using a sensor* section, page 34, for more information on variables.

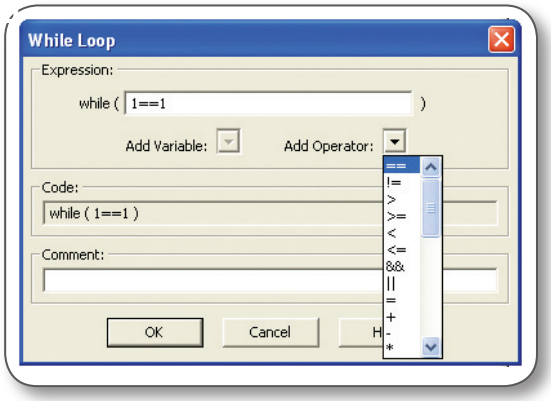


3. In the "Function Block" window, under the "Program Flow" heading, find the "While Loop" icon. Left-click on the icon and drag it into the program window and drop it between the begin and end icons.



4. The "While Loop" configuration menu will appear. Set the condition of the while loop to be "1==1". This has the same effect as using a variable set to 1, like in the sample bump sensor code. Click "OK".

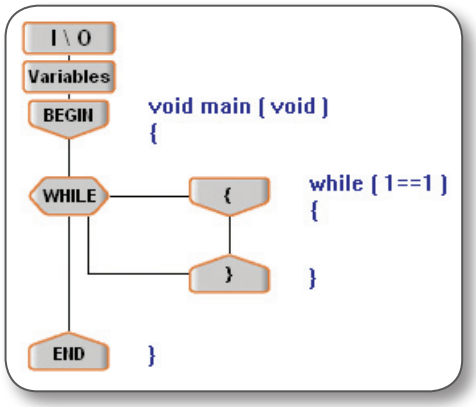
The "While Loop" will loop for an infinite amount of time.



06/30/05

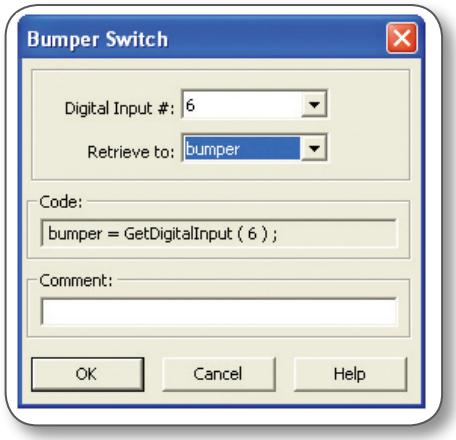
program three: using motors and sensors together, continued

5 Your program should now look like this.



6 In the "Function Block" window, under the "Inputs" Heading, find the "Bumper Switch" icon and drag this into the program window between the { and } brackets of the "While Loop". Set the "Digital Input #" to 6 (where the front bumper switch is plugged in, see page 33 for setup details), and the "Retrieve to" to the variable you created, "bumper". Click OK when you are done.

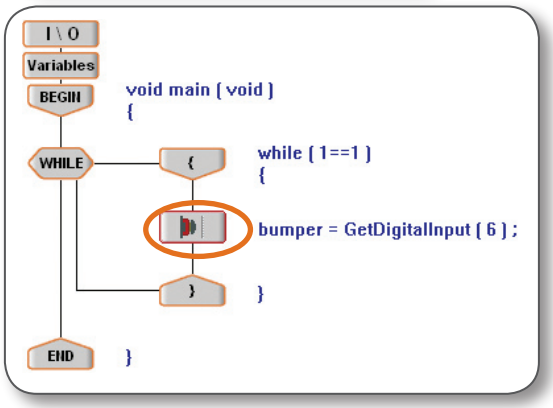
You have now set the variable "bumper" to be updated with the current value of the bump sensor each time the loop is run.



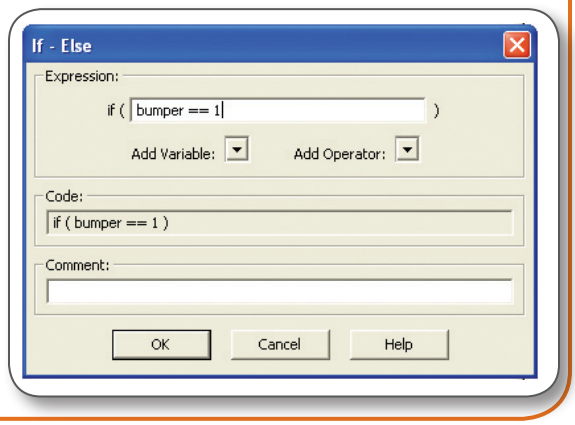
7 If the bump sensor is not pushed in, then you want the motors to continue moving the robot forward. If the bump sensor is pushed in, then you want the robot to stop.

Since we are storing the bumper sensor's pushed/not-pushed state in the "bumper" variable, that is where the program must look to see whether it is being pushed. If the sensor is pushed, "bumper" will have a value of 0. If it is not pushed, "bumper" will be equal to 1.

We will use an if-else statement to perform this check.

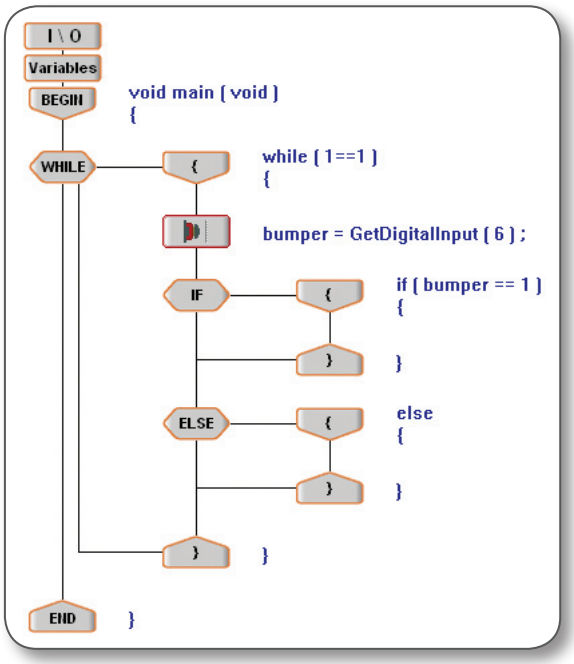


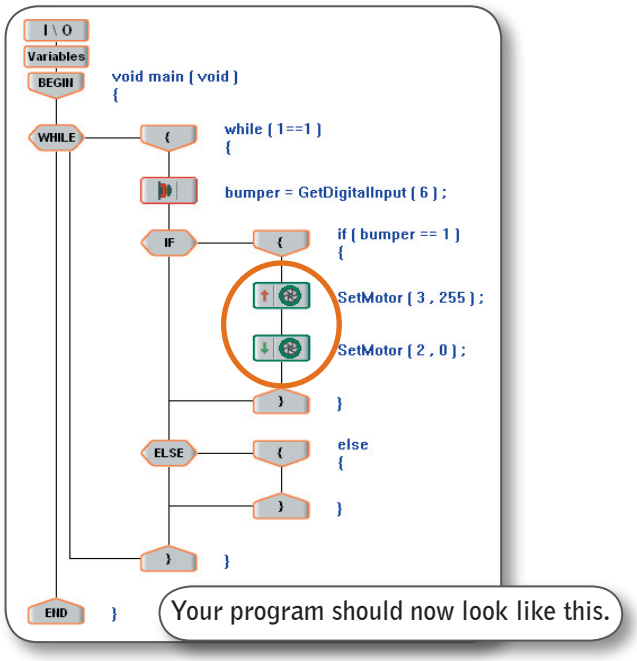
8 In the "Function Block" window, under the "Program Flow" heading, drag the "If-else" block into the "While Loop" underneath the "Bumper Switch" icon. In the "If-else" configuration window, the first block asks you to define the condition (just like a while loop). First, click the small arrow next to "Add Variable" and select "Bumper". Then, click the arrow next to "Add Operator" and select the "==" operation. After the "==" sign, you will enter the number 1. The if-else condition will now check whether the variable "bumper" is equal to 1. Click OK to continue.



program three: using motors and sensors together, continued

8 Your code should look like the program on the left. Now, recall that if the bump sensor is not being pressed in (that is, bumper is equal to 1), we want the robot to move forward. You already know which combination of icons makes your robot move forward (refer to *program two* if you need a reminder). Drag in two motor modules between the brackets under the "if" statement. Set them up so that your robot will drive forward. Be sure to place the move-forward motor commands in the section immediately after "if (bumper==1)" because you want them to execute when bumper is equal to 1.





Your program should now look like this.

9 The other half of this program's behavior is that if the bumper is being pressed in, we want the motors to stop. Drag in two motor modules between the brackets under the "else" statement. Configure them so that they form a stopping behavior (see *program two*).

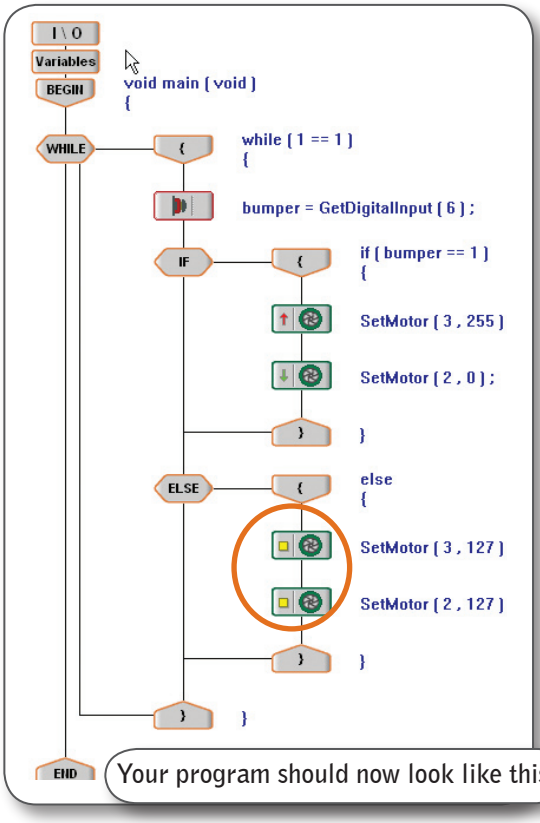
10 Check your code to make sure it looks like the code on the right. Compile and download your code.

Refer to the *programming sequence* section for detailed steps on compiling and downloading

11 Test your code. Your robot should now drive forward until the front bump sensor is hit, at which point the robot will stop. If you move the robot so that the front bump sensor is no longer pressed, it will once again move forward.

12 Save your program as "intro program three". We will be revisiting this program soon! For a refresher on how to save your program, see *tips for saving programs* on page 26.

Congratulations, you have integrated sensor feedback into your robot behavior!



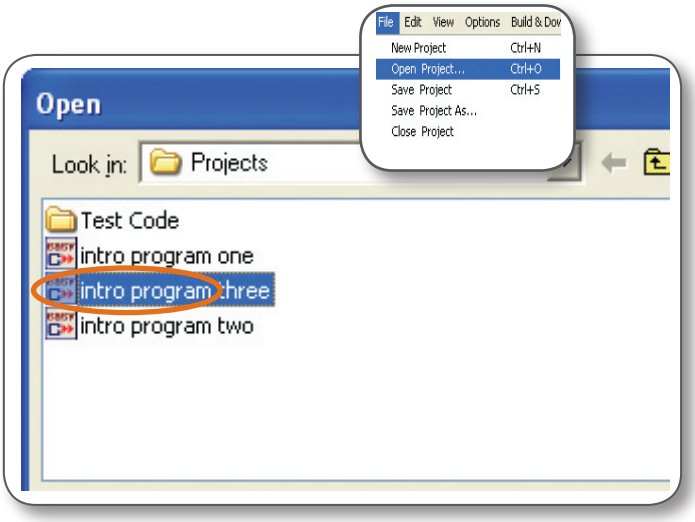
Your program should now look like this.

06/30/05

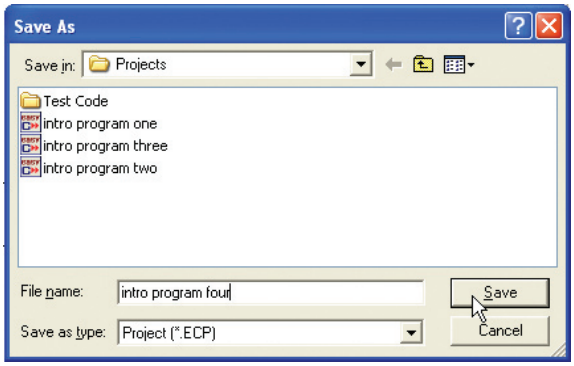
program four: reversing and turning

In this section, you will enhance the code from program three by making it back up, turn, and then continue on its way when the bumper sensor is hit, rather than just stopping.

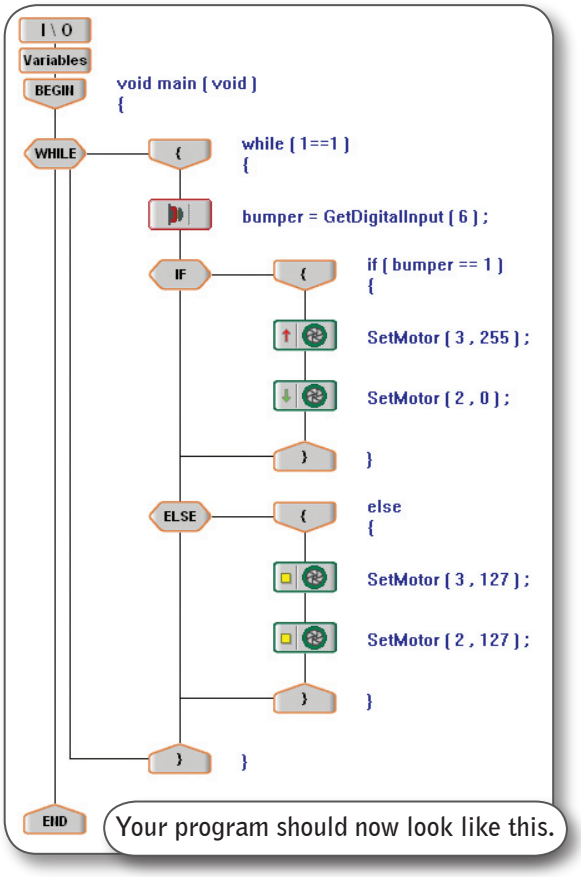
1 Open your saved program three file. Select "Open Project" from the "File" menu, then click on INTRO PROGRAM THREE and click "Open".



2 Before editing the code, save the code as "intro program four".



For a refresher on how to save your program, see *tips for saving programs* on page 26.



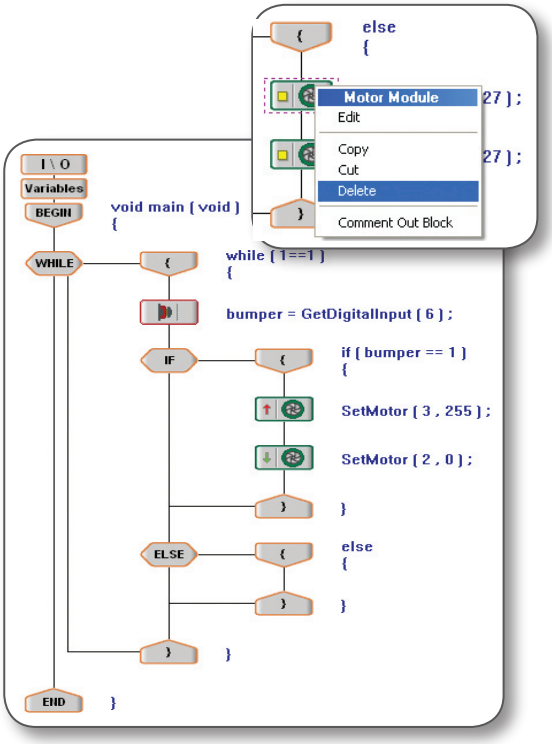
Your program should now look like this.

program four: reversing and turning, continued

3 The current programmed behavor is: if the bump sensor is not being pressed in, the motors turn on; when the bump sensor is pressed in, the motors turn off.

We will edit this program so that instead of stopping when the bump sensor is pushed, your robot will back up, turn right, and then resume moving forward. This means that the new turn-backup code will be replacing the old stop code icons in the else statement.

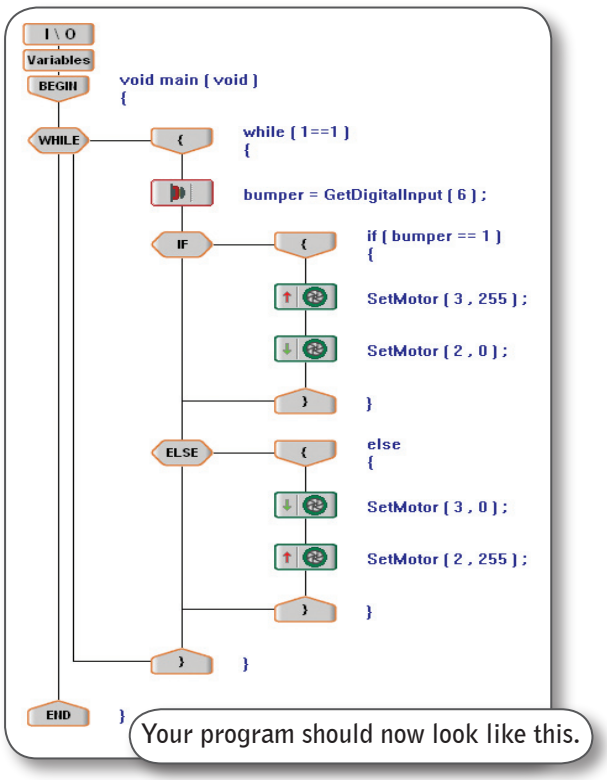
Start by clearing away the old code. Delete both of the Motor Module commands in the else statement. Right-click the first icon, then select delete from the menu that appears. A screen will appear asking if you are sure; click yes. Repeat for the second motor block.



4 Next, we want to start building the new behavior into the else statement. Begin with the first action, backing up (the idea is to move your robot clear of the obstacle it has encountered).

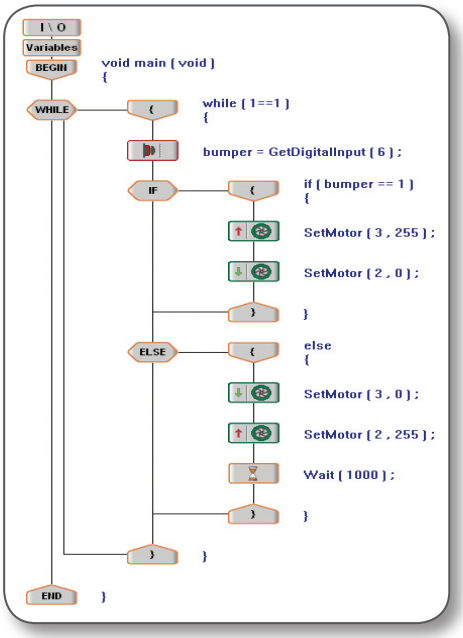
To do this, drag in two "Motor module" icons between the brackets of the else statement. Configure them so the robot starts moving in reverse.

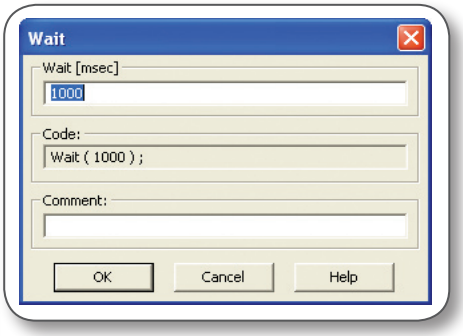
See *using the motors* on page 27 for the motor directions



program four: reversing and turning, continued

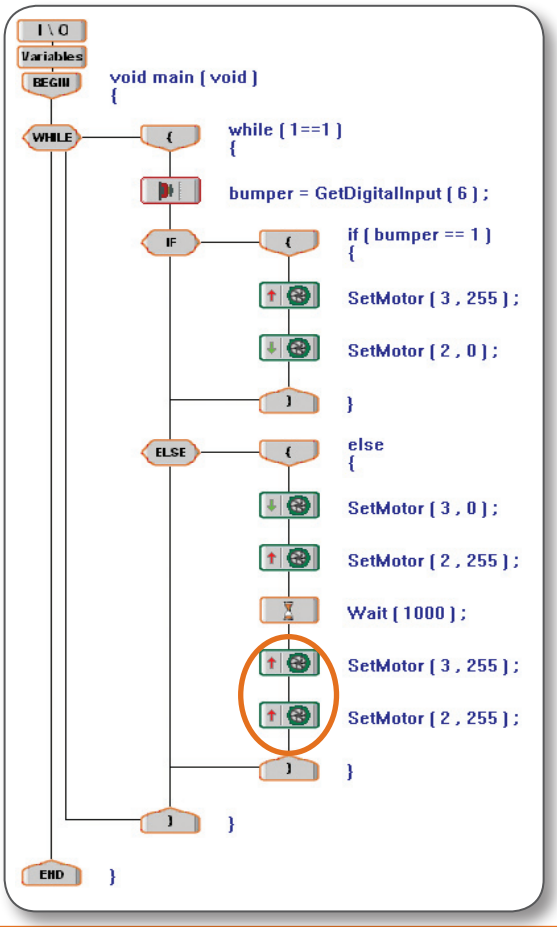
5 Set the amount of time for the robot to move with a "Wait" icon like you did in program two (page 27). Drag a "Wait" icon into the else statement under the two "Motor Module" icons. In the "Wait" configuration window, set the "Wait [msec]" to 1000 (one second.)





6 To make your robot turn right, drag two "Motor Module" icons below the wait icon, inside the else statement. Configure them so the robot turns right.

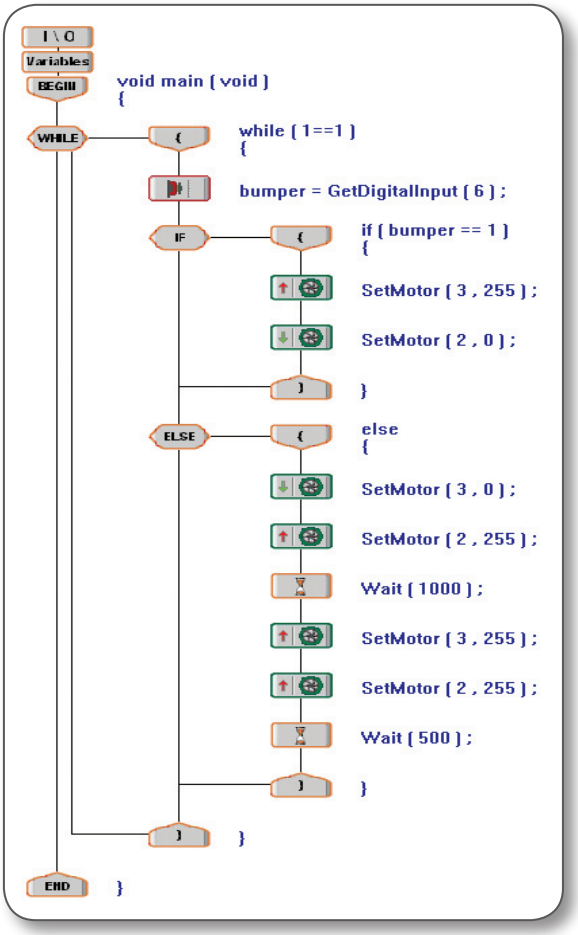
See *using the motors* on page 27 for the motor directions



program four: reversing and turning, continued

7 To specify the amount of time your robot will turn right, drag a "Wait" icon into the else statement under the two "Motor Module" icons. In the "Wait" configuration window, set the "Wait [msec]" to 500 (which is 500ms, or half a second.) This should make your robot turn approximately ninety degrees.

Note: The amount of turning produced by this program will vary from robot to robot. You may have to tweak the timing for your robot to turn away from the wall.



Your code should now look like the program above

8 Compile and Download your program.
Refer to the *programming sequence* section for detailed steps on compiling and downloading.

9 Test your code. Your robot should drive forward until the front bump sensor is hit, at which point the robot will back up, turn right, and then resume forward motion.

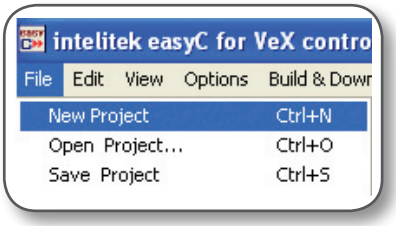
10 Be sure to save changes to your code. Refer to *tips for saving programs* on page 26.

Congratulations! Your robot now performs an intelligent autonomous behavior! Let your robot run for a while to see how long it can continue roaming around a room. Identify areas for improvement in the design, and construct and program an improved version.

program five: using the radio control transmitter

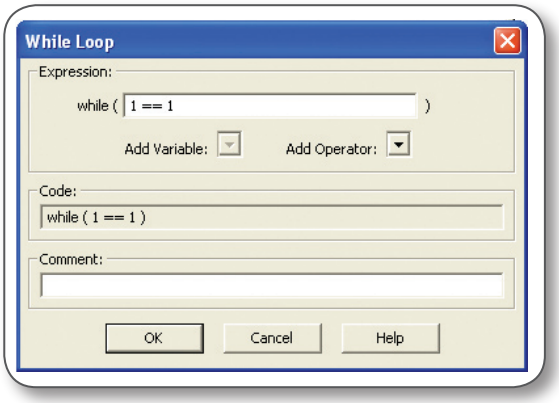
This section will explain how to incorporate Radio Control into your programs. Please make sure your RF receiver module is plugged into the RX1 port of your Vex Micro Controller. The rest of your robot should still be configured according to the instructions on page 13.

1. Start a new project in easyC by clicking "New Project" in the "File" menu.



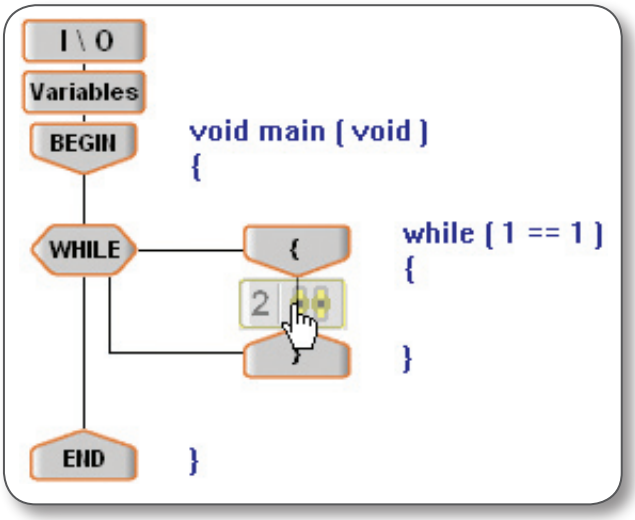
2. In the "Function Block" window, under the "Program Flow" heading, find the "While Loop" icon. Drag this into the program window between the Begin and End icons.

3. Set the condition of the "While Loop" as "1==1" in order to create a continuous loop. Click OK.



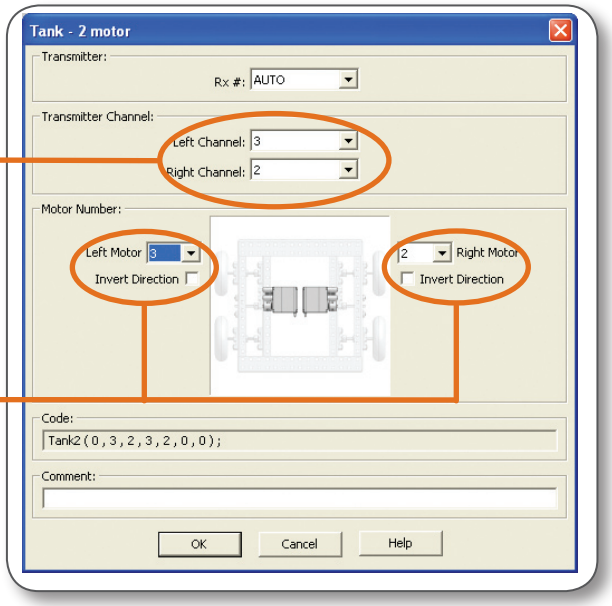
4. In the "Function Block" window, under the "RC Control" heading, find the "Tank - 2 motor" icon. Drag the "Tank - 2 motor" icon into the "While" loop (drop it between the { and } icons as shown). This icon causes the robot to wait for a signal from the radio control transmitter.

Note that the RC Icons only work when they are inside a loop. The icon must be executed repeatedly by the program in order to provide continuous control. Otherwise it will only give you radio control for an invisibly brief moment, then move on.



program five: using the radio control transmitter, continued

5 The "Tank - 2 Motor" configuration window will appear. Leave the RX# set to auto.

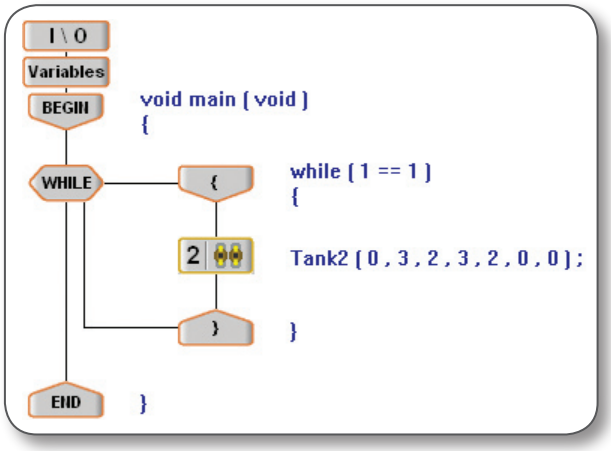


6 Set the left channel to "3", indicating the vertical axis of the left joystick on the transmitter will be considered the "left" control. Set the right channel to "2", indicating the vertical axis of the right joystick on the transmitter will be considered the "right" control.

7 In the "Motor Number" section, set "Left Motor" to "3" and "Right Motor" to "2". This specifies which motors are being controlled, by the corresponding joysticks selected in step 6. The motor numbers are determined by your current motor set up, which is described in the *motor and sensor setup* section on page 38.

8 Click OK to continue. The icon should appear between the "Begin" and "End" icons with "Tank2(0,3,2,3,2);" beside it.

Note: "Tank2" means that you chose the Tank-style controls with a two motor setup. The first number, "0", shows that the program will automatically use the RX port that has a signal. The first 3 and 2 following that show the transmitter channels. The second pair of numbers, 3 and 2, show which port you set the left and right motors to, respectively.



10 Compile and download your code. (Refer to the "Programming Sequence" section for detailed steps on compiling and downloading.)

11 Test your code. Your robot should be controllable using the radio control transmitter. The right joystick will control the right motor and the left joystick will control the left motor.

12 Save your program as "intro program five". For a refresher on how to save your program, see *tips for saving programs* on page 26.

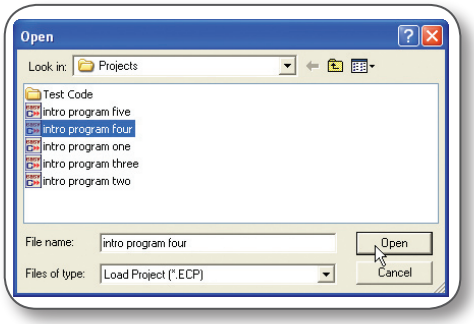
Congratulations, you have programmed your robot sucessfully!!

06/30/05

program six: combining autonomous and radio control

In this section, we will explain how remote control code can be combined with autonomous code. In this example we will adjust the code from program four to include remote control similar to that in program five.

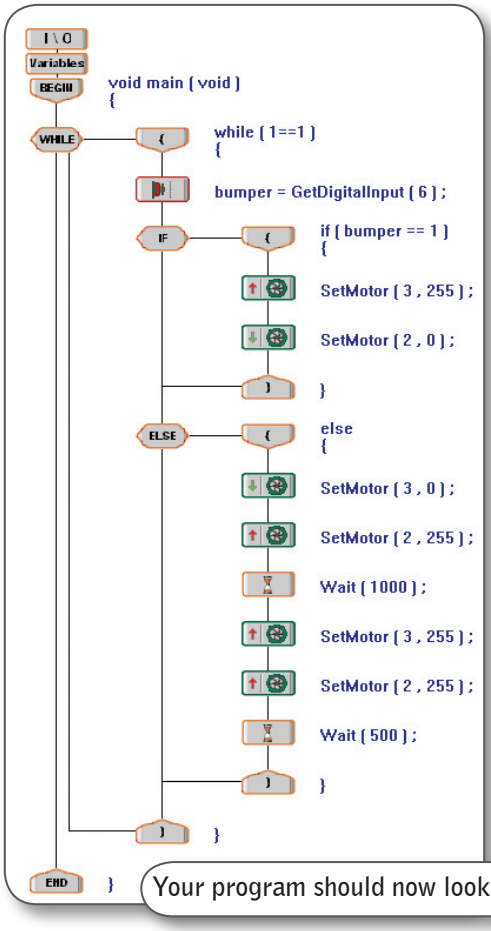
1 Open your saved program four. Select "Open Project" from the "File" menu, then click on INTRO PROGRAM FOUR.ECP (or whatever you saved it as) and click "Open".



Refer to the *tips for saving programs* section, page 26, if you are unsure of how to do this.

Recall that in this program, the motors switched on autonomously if the bump sensor was not being pressed in. When the sensor was hit, the robot would back up, turn, then continue on its way.

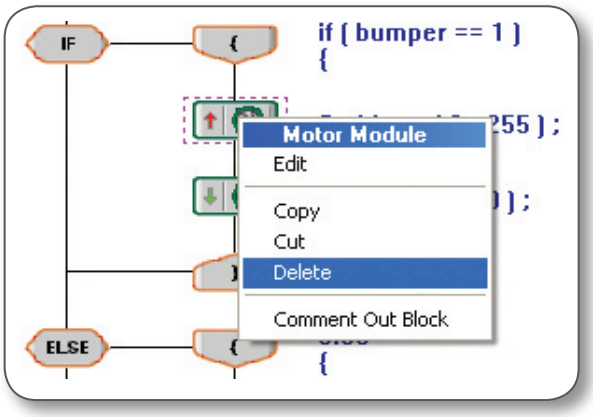
Now, you will modify this code so that instead of driving forward, you will have control of the motors via the radio control transmitter unless the front bump sensor is pressed. This means that code to enable radio control will replace the old move-forward icons in the if-else statement.

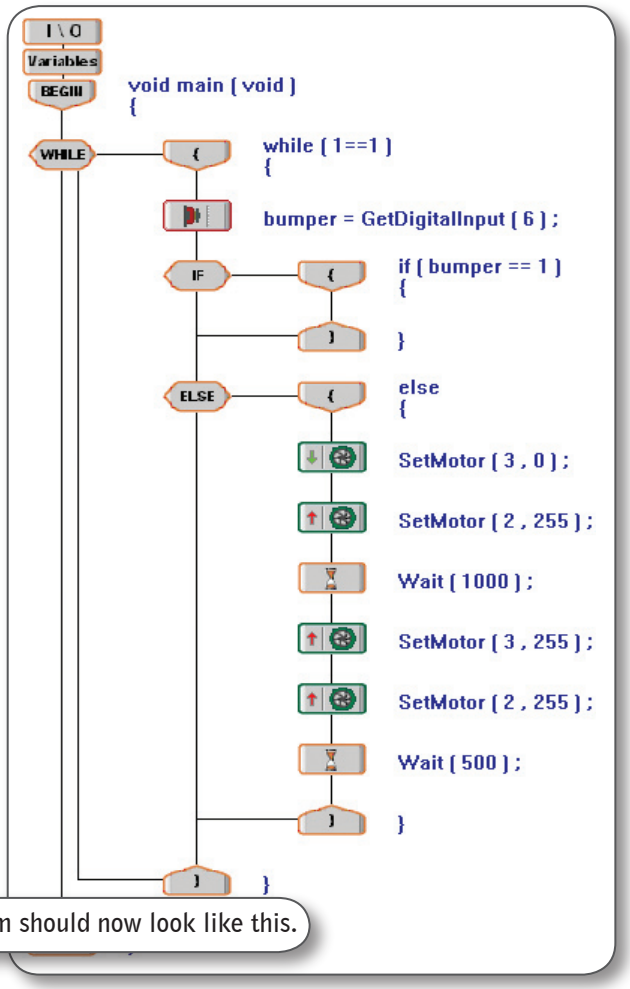


Your program should now look like this.

program six: combining autonomous and radio control, continued

2 Start by clearing away the old code. Delete both of the "Motor Module" commands in the if statement. Right-click the first icon, then select delete from the menu that appears. A screen will appear asking if you are sure; click yes. Repeat for the second motor block.

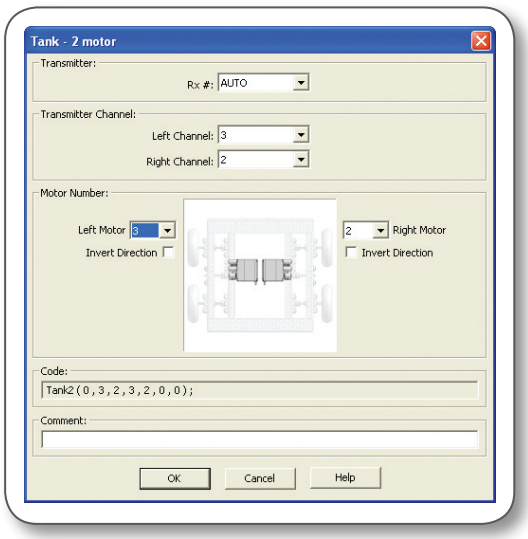




Your program should now look like this.

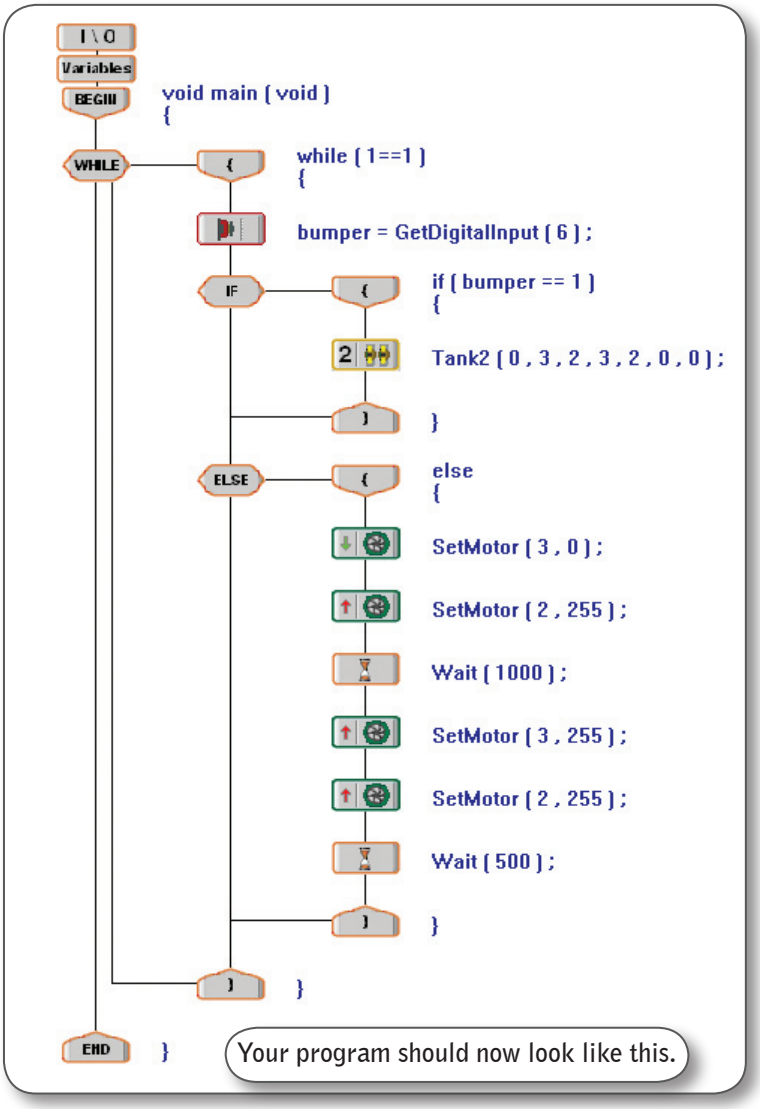
3 Next, you want to put tank control identical to the code you used in program five into the if statement. To do this, drag the "Tank-2 motor" icon from the "Function Blocks" window into the program area and drop it between the { and } of the if-statement. Set up the "Tank 2-motor" configuration window as shown.

For more information about putting tank control into a program, refer to *program five* and the *using radio control* section.



06/30/05

program six: combining autonomous and radio control, continued



4. Compile and Download your code. Refer to the *programming sequence* section for detailed steps on compiling and downloading.

5. Test your code. Your robot should now be controlled by the radio control transmitter, as long as the front bump switch is not pressed in.

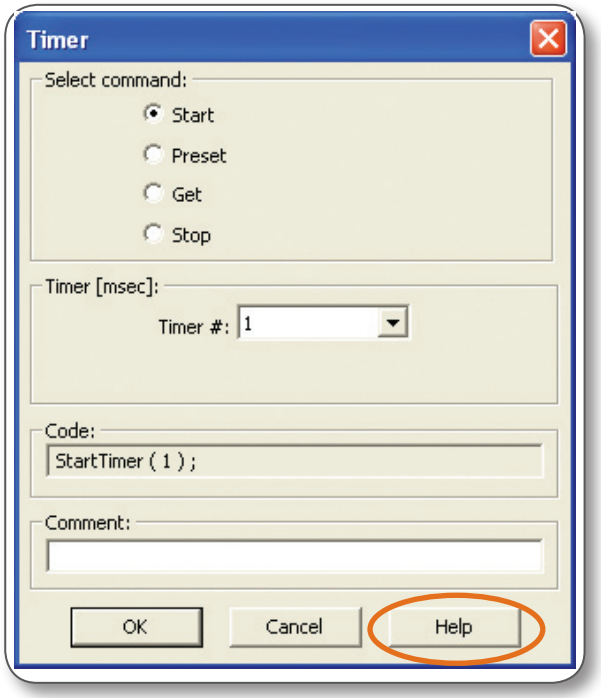
6. Save your program as "intro program six".
For a refresher on how to save your program, see *tips for saving programs* on page 26.

Congratulations! You have successfully combined autonomous and radio control code to give you a robot that can be manually controlled, but has a built-in safety mechanism that backs the robot up and turns it away from an obstacle when it detects a collision.

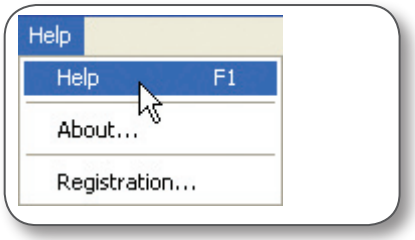
additional help

In addition to the tutorials and instructions provided here in the programming guide, easyC has its own set of help files that may help you find the answers you seek.

1. Help on any specific function block (like a "Wait" block or a "Timer" block) can be found by dragging the icon you are curious about into the program window. The configuration window should appear and there will be a "Help" button in the lower right hand corner of the window.



2. For more general information about using easyC, try the help menu on the menu bar at the top of the screen.



troubleshooting

There are different problems that you may encounter while programming with easyC. Here are some of the more commonly encountered ones, and some possible solutions.

Programming Strategies:
Describes strategies to help you program more efficiently.

Compilation Errors:
Covers errors in your code.

Other Errors:
Covers download and access right errors.

Download Errors:
Describes errors when downloading your code to the robot.

Access Right Error:
Covers errors that occur when you are not an administrator.

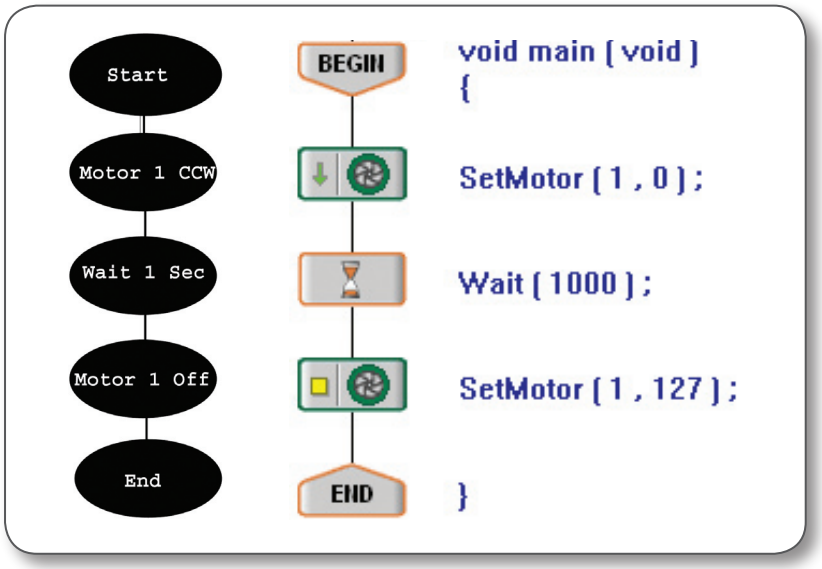
NOTE : If you would like to download the default code that accompanies your robot, refer to page 22.

programming strategies

This section will outline useful coding habits that may help you program more effectively.

1. Test each section of your program to make sure each one works individually. The program as a whole can't work if the parts don't.
2. Build your program step by step and test at intervals. Never write out the whole program at once.
3. Use "Wait" and "Print to Screen" commands at strategic points in your program. This will help you identify where your errors occur.
4. Use comments to document what important lines in your code are doing. You aren't going to remember every single line of code when you come back to your program in a week, and it helps other programmers on your team to understand how your code works.
5. When using loops or if (-else) statements, use the drop-down menus to help write the condition.
6. Don't use the "User Code" icon unless you have experience in programming. Using the "User Code" icon allows you to write your own code that the easyC software cannot help you correct.
7. easyC can not identify all of the errors you will make, because your incorrect code may still look like a legitimate program. Programs may compile correctly, yet will not work as you want them to.
8. If you are using sensors, be sure you connect them to an input port of the correct type. For more on how to change this refer to the *intro to sensors* section (page 33).
9. Save your code often. Also save your code under a different name before making major changes so you can revert back to an earilier version if needed.

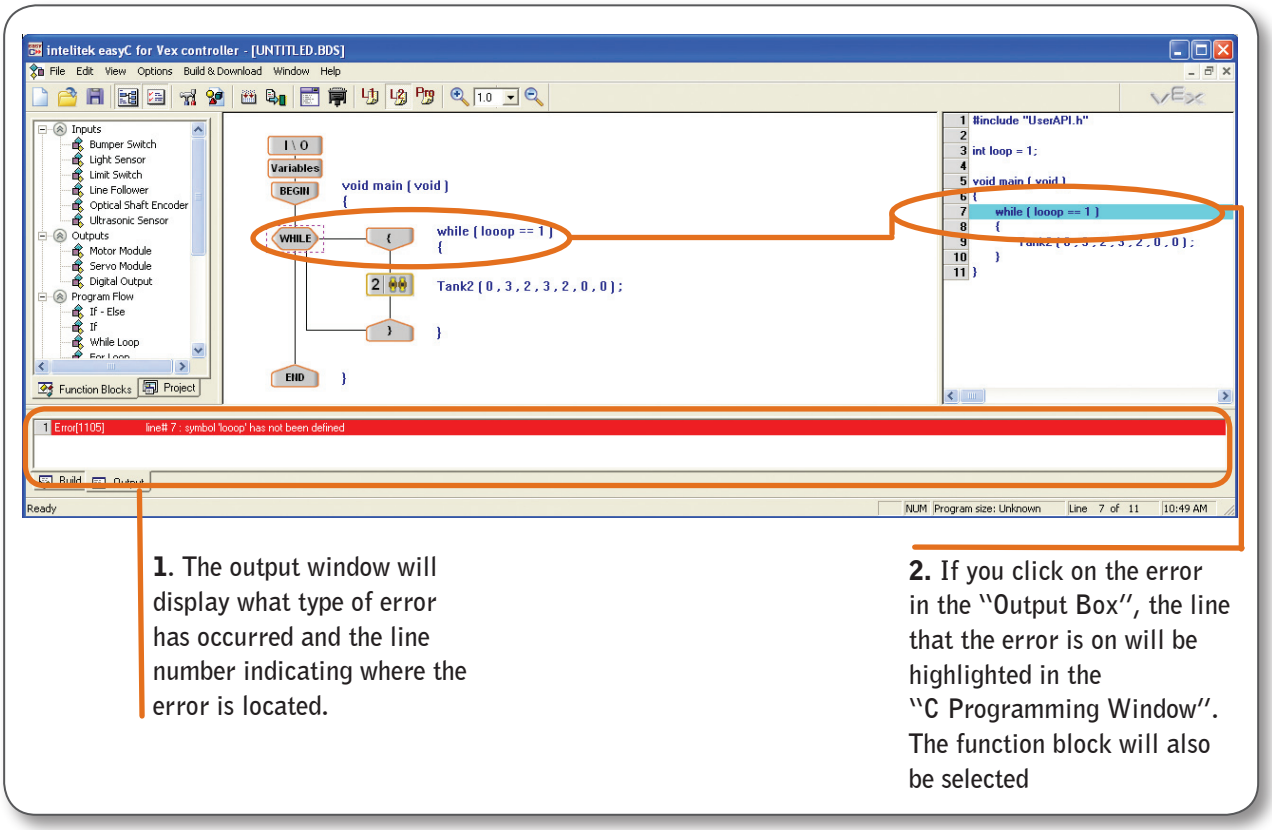
Before beginning to code a program, identify the behaviors you will need, and how they will need to be arranged. You can do this by simply writing out the steps or using a flow chart like the one below.



compilation errors

Compilation errors are errors in your code that prevent the Vex Micro Controller from understanding the program. This section will teach you how to use easyC to find errors in your code.

If you click on the "Build and Download" button and you have compilation errors in your code, you will receive this error message.



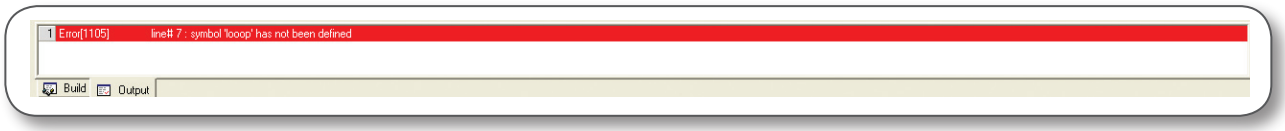
1. The output window will display what type of error has occurred and the line number indicating where the error is located.

2. If you click on the error in the "Output Box", the line that the error is on will be highlighted in the "C Programming Window". The function block will also be selected

compilation errors, continued

Compilation errors will be displayed in the "Output Window" according to the following format:

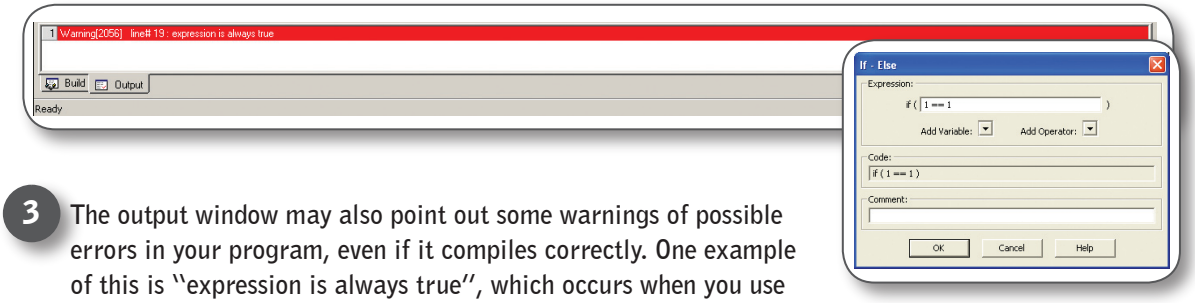
Error [error number] line# (the line number of the error) : description of error



There are several common errors displayed in the "output window":

1. "Symbol has not been defined": This occurs when you misspell a variable name or if you try to use a variable that has not been defined in the Variable Window at the top of the program.

2. "Syntax error" : This occurs most frequently when you do not complete the condition in an if statement or loop. For example, in "if(loop ==)", the "loop == " condition is not a complete statement.



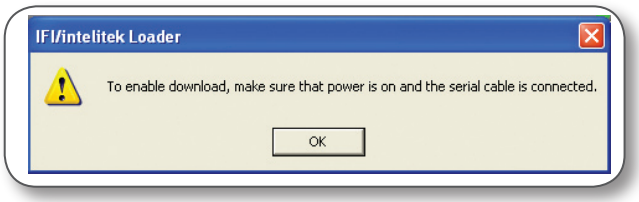
3. The output window may also point out some warnings of possible errors in your program, even if it compiles correctly. One example of this is "expression is always true", which occurs when you use an expression that is always "true" in an "if-else" conditional statement (the program warns you that this is suspicious – after all, why would you want an if-else statement that never used the else block?)

other errors

1. Download errors

Download errors occur when you are attempting to download code to your robot. These errors can occur because your hardware is not set up properly or because your software is not configured properly with your hardware. This section will teach you what the different download errors mean and how to correct them.

If you are downloading code and you receive this error message, try the following steps to correct the error.



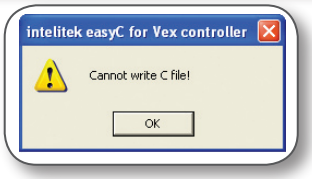
1. Check to make sure your hardware is set up according to the programming hardware page (page 13).
2. Make sure your robot is on.
3. Make sure you have properly installed the USB-to serial driver and your COM port is set properly (refer to *installing the easyC programming software,* page 13).

If the loader freezes up and then closes in the middle of downloading, try the following steps to correct the error:

1. Check to make sure your robot was not turned off during download. Check your battery levels.
2. Be sure all cables are plugged in.

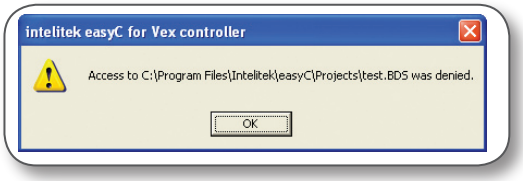
2. Access rights errors

Access rights errors will occur if you are not set up as an administator on your computer.



This error message will appear if you are not an administrator and you attempt to compile a program.

This error message will appear if you are not an administrator and you attempt to save a file in a restricted location.



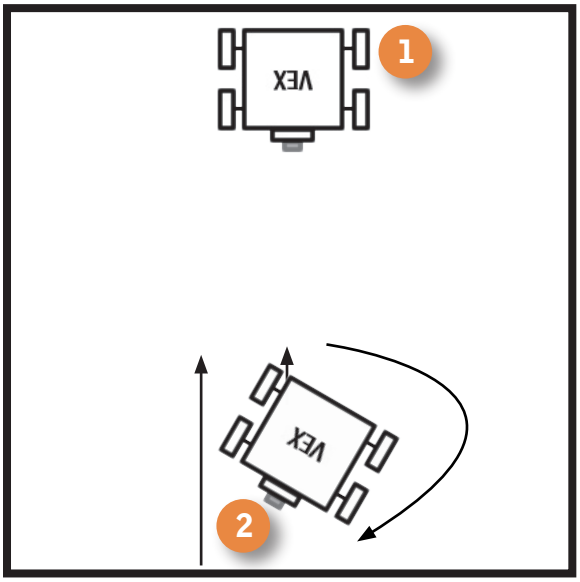
To solve this problem, talk to your system administrator about getting your user privileges upgraded to include file write access to the folder on the computer where easyC is installed. This is "C:\Program Files\Intelitek" on most machines.

keeping count

Build and program a wandering robot that will run straight until it hits a wall, then back up and turn, all the while keeping track of the number of walls it has hit. After hitting a wall for the third time, the robot must come to a complete stop.

The code solution to the challenge can be downloaded from www.Vexrobotics.com

1. The robot must begin facing a wall.
2. Once it hits a wall, it must reverse for 1 second, then turn or pivot right, and then resume traveling forward.
3. The program should stop after the third wall collision (it can back up and turn, but should not move forward).



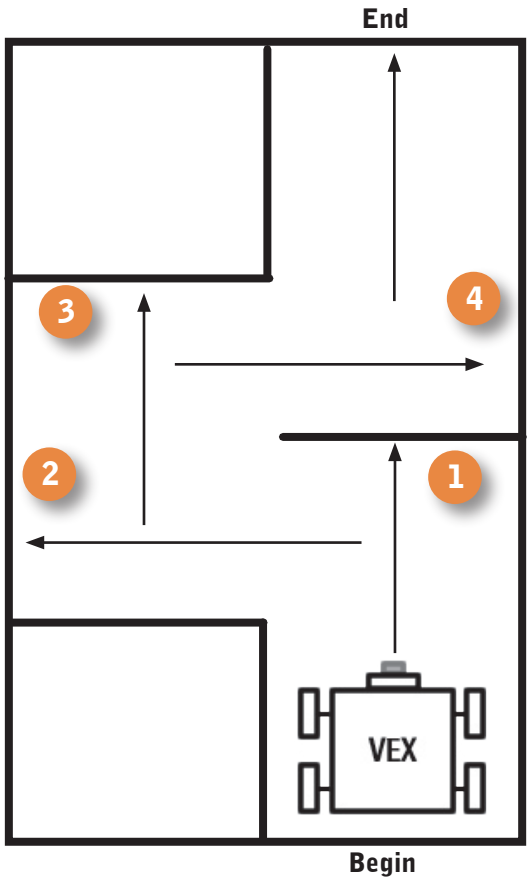
Use this space to draw a flowchart and write your solution:

headache!

Use a bumper switch to help you navigate a maze by "feeling" your way through. You will be able to see the maze ahead of time to plan your route, but your robot must use the bumper switch sensor to detect walls.

The code solution to the challenge can be downloaded from www.Vexrobotics.com

1. You will be able to see the maze ahead of time to plan a route.
2. Your robot must be able to move from the starting area to the finish area without human assistance once turned on.
3. The robot must use wall collisions to guide it through the maze. It cannot navigate by timed movements alone.
4. Hint: Like most long sequences of behaviors, this one is easier to program if you write and test your program one segment at a time.



Use this space to draw a flowchart and write down ideas: