

## **Basic Programming Tips**

### **Tip #1: Avoid confusion by programming in small steps**

Novice programmers frequently attempt too much at once with their programs. This practice tends to leave people feeling confused, overwhelmed, and often at a loss for what is really going on in their program. A better practice is to start out using small steps. This is less confusing and less frustrating when troubleshooting your programs.

**Example:** The team wants its robot to drive forward three feet, turn 90 degrees to the left, drive until it sees a black line, stop and then beep. The team tries to program all these steps at once and then test the robot. Needless to say, the desired result is not achieved on the first try. A better way to go about programming this is to use a step-by-step method for programming.

- Step 1:** Break the problem up into the individual programming steps required. In the above example the program would require 5 steps to complete the program.
- Step 2:** Program the robot to drive forward three feet. Test this step to make sure this achieves the desired result. Then and only then, move on to step 3.
- Step 3:** The 90-degree left turn. Again, test this step to make sure it achieves the desired result before moving on to step 4.

Following this method you should achieve great results, saving time while attaining a better understanding of your program. To use these small steps easily in your program, be sure to put the step in a “My Block” (or Sub VI for Robolab™). “My Block” or Sub VI for Robolab™ allows you to easily reuse these small steps that you have taken the time to create. Please check out the section on “My Blocks” if you are unsure of them.

### **Tip #2: Take time to learn the basics**

LEGO programming is not as difficult to learn as many other programming languages, such as Java and C++ . Learning RIS or Robolab™ will take a little effort. Robolab™ users should take the time to read the Robolab™ programming manual. Download the programming manual if you need to. It will pay off a hundredfold. You can find the Robolab™ manual at <http://www.cceo.tufts.edu/robolabatceeo/Resources/documentation/ROBOLAB%20Reference%20Guide.pdf>

MINDSTORMS RIS does not have a manual for programming. Instead, use the training challenges on the RIS software CD.

Practice Programming. Consider programming simple tasks for your robot, being sure to include these tasks:

- Getting your robot to drive forward.
- Turning your robot left and right.
- Driving your robot in reverse.
- If you are using a motorized extension of any kind practice getting that mechanism to operate as you envision.

### Tip #3: Save Often.

It is important to remember to save often! Losing five to ten minutes worth of work is less painful than losing three to eight hours work time. Save often should be your mantra. At tournaments, teams sometimes feel that if they just had five more minutes they could have done much more. Saving often will ensure that you will not lose the precious time that you have to program. So remember your mantra, SAVE, SAVE, SAVE -- often.

### Tip #4: Stay away from Big Blocks

Robolab™ users may ignore this tip, as Big Blocks are not a part of Robolab™ programming.

In MINDSTORMS™ Robotic Invention System 2.0, LEGO MINDSTORMS™ introduced the Big Blocks concept. “

“Big Blocks” - Green	“My Blocks” - Yellow
<p>Big Blocks are:</p> <ul style="list-style-type: none"><li>• Pre-made “My Blocks” for the robot designs detailed in the Constructopedia.</li><li>• Do different things for each robot design.</li><li>• Meant to help the home user get a grasp on programming the pre-designed robots very quickly.</li></ul> <p>Big Blocks use timing; therefore the robot will slightly change what it does as the batteries drain.</p> <p>To open them up, press the gray button to see the mini-programs. These programs can serve as great guides to making your own “My Blocks.”</p>	<p>The user names them.</p> <p>The user can write specialized commands made up of many small blocks that do not have to rely on timing, but can rely on sensors.</p>

### Tip #5: Use “My Blocks,” or \***“Sub VI”** in Robolab™, for repeatability.

“My Blocks” are a good feature to use in any program because they let a user write specialized commands made up of many small blocks. Use this feature to guarantee repeatability in your program. Think of this as writing a mini-program, sometimes called a subroutine, that you can use many times in your program.

Each time a “My Block” is called, it will run the same command every time.

**Example:** If you write the perfect 90-degree turn for your robot, you can use it any number of times without rewriting it.

**But:** When dealing with “My Blocks,” remember that many different programs can use the same “My Blocks.” If you change a “My Block” in one program, it will change all instances of that “My Block” in that program and other programs.

**Solve:** Instead of changing the “My Block, make a new “My Block” and label it with a slightly different name. Copy the small bricks from the existing “My Block” and paste them into the new “My Block”.

\* Please note that the concepts talked about that apply to “My Blocks” also apply to “Sub VI” in Robolab™.

### **Tip #6: Use Sensors**

Sensors are your best method of maintaining consistency with the same robot across a period of time. A sensor will still accurately give its reading even when the batteries in the RCX are running lower. Programs that do not use sensors, touch, light or rotation, must rely on timing. The timing on your robot becomes less accurate as your robot's batteries drain. Use sensors; don't let your program degrade with the quality of the batteries on your RCX.

**Example:** Robot takes three seconds to drive to a black line on full battery power. The same robot at half battery power could take five seconds to drive to the same point. But, if you use a light sensor to sense the black line, it will consistently find it no matter how long it takes the robot to drive to the line.

### **Tip #7: Stay Organized**

Try to keep your program neat and readable. When saving the program (RIS only), write as many notes in the logbook as you need to help you remember what your program does. Using the logbook might seem unimportant, but it will help when using programs from year to year and user to user.

Robolab™ has an even better method of keeping notes. Robolab™ allows you to make text boxes anywhere in your program. Just think about how useful it is to be able to write a note to yourself about what your program is supposed to be doing at this point in your program. Even better would be to write a note to yourself letting you know that up to this part of your program your program is tested and works properly. This is very useful especially in "Sub VI"s.

### **Tip #8: Don't be scared of the Help screens**

Both Robolab™ and RIS contain Help menus. Use them to determine the function of each particular brick.

### **Tip #9: Determine which software is best for you**

Things that you will need to check for include:

- Which program will run on your computer's operating system?
- Which program is easier to learn?
- Which program is more like a traditional programming language?

## **Frequently asked Questions**

The following is a list of frequently asked programming questions with possible solutions.

---

### **How can I print out my program?**

To print in RIS follow these steps:

1. On the keyboard, notice a key labeled "Print Screen..
2. With your program showing on the screen, hit that key.
3. Open MS Word (or your preferred text editing program).
4. Click "paste." This should paste your program as a screen shot (picture of the screen).

5. Be sure to save the MS Word document.
6. Repeat until your entire program is in the document.

Please note: Robolab™ allows you to print your programs in an easier fashion. In Robolab™, Select File -> Print.

### **The sensors on my robot are not working.**

Make sure the sensor is expecting feedback from the correct port. The RIS software will take the next available port in the program, not necessarily the correct one. Make sure the sensor is attached properly and to the correct port on the RCX. Did you connect the sensor back to a different port than when you last used this program?

### **Sensor value doesn't show in the RCX view window.**

Make sure to download a program into the RCX that contains a sensor watcher and run it. The sensor watcher must be set to the type of sensor and to the port into which the sensor is plugged. (Basically, you are telling the robot what type of sensor input to expect from which port.)

Verify that the correct program slot is selected. Run the program. Make sure that in "view mode" the arrow is pointing to the correct sensor port. The RCX will then read the correct information for that sensor watcher on that port.

### **I can't access program slot number one.**

In RIS: Go into the options screen. Click on the "advanced" tab and select to unlock program slot #1.

In Robolab™: Go into the Administrator's section. Click on the "RCX settings" tab on the bottom of the screen. Click to unlock RCX program slots 1&2.

### **The RCX lost all my programs after I changed the batteries.**

Download the firmware again. This will reinstall the five default programs. If you changed the RCX by unlocking any program slots, you will need to repeat that process after you reinstall the firmware. After this is complete, download your programs into the RCX.

### **My robot keeps spinning in circles after I programmed it to go straight.**

Check that the program uses a second set direction command to adjust the motors so they are going forward. If that does not work, using the default program in slot number one in the RCX, test that the robot moves in a forward direction as planned. If this is not the case, adjust the wires connected to the motor, spinning the connection to the reverse direction on one end of the wire, until it drives forward.

### **I changed a "My Command" (or "Sub VI" in Robolab™) in one of my programs. After I downloaded my programs again, none of them works properly.**

A "My Command," or "Sub VI" in Robolab™, is a miniature program that is accessible by all of the programs that you write.

**Beware:** If you change a "My Command" or "Sub VI" in Robolab™ in one program, *you are actually changing it in every program it is used.* To prevent this, make a new "My Block" (or "Sub VI" in Robolab™) with a

slightly different name, copy the existing “My Block” or “Sub VI” and paste it into the new one.