

Movement

Moving Forward Code Dissection

Now that you understand the steps needed to download a program, which you learned through downloading a sample program, let's take a step back and figure out what all of this "code" is really doing.

Here on the main screen we have several lines of code. Let's walk through what each of these commands "does".

```

1 task main()
2 {
3
4     motor[motorC] = 100;
5     wait1Msec(3000);
6
7 }
  
```

task main()
This line creates a task called main, which contains the behaviors that we want the robot to do.

task main() marks the beginning of a structure.

{ body }
{ and } "braces" surround the body of the structure. The lines between the braces tell the program what commands to follow as part of the main task.

As you know, the code currently tells the robot to move in a circle. More literally, it tells the robot to move "Motor C" forward for 3 seconds. Moving only one motor, or wheel, will make your robot go in circles. The details of each command are as follows:

motor[] command

The motor[] command tells the robot to set a motor to run at a given power level. The example below (taken from the program you ran) sets motor C to run at 100% power forward. Note that every command in ROBOTC must end with a semicolon, just as every English statement must end with a period.

Example:

```
motor[motorC] = 100;
```

wait1Msec() command

The command "wait1Msec" tells the robot to wait, for the given time in milliseconds. The number within the parenthesis is the number of milliseconds that you want the robot to wait. 3000 milliseconds is equal to 3 seconds, so the robot moves for 3 seconds.

Example:

```
wait1Msec(3000);
```

Movement

Moving Forward **Code Dissection** (cont.)

In this lesson, you will modify the existing program code to create a Moving Forward behavior with the robot.

- Before making any changes, save the program with a new name. Go to "File", "Save As", and rename this program to "Labyrinth".

1a. Save program As...
Select File > Save As... to save your program under a new name.

1b. Browse to an appropriate folder
Browse to or create a folder (on your desktop, in your documents folder, etc.) that you will use to store your programs.

1c. Rename program
Give this program the new name "Labyrinth".

1d. Save
Click Save.

- Add a new line after the first motor command.

```

1 task main()
2 {
3
4     motor[motorC] = 100;
5      
6     wait1Msec(3000);
7
8 }
```

- 2. Add this space**
This is where we will add the second motor command in the next step.

Movement

Moving Forward **Code Dissection** (cont.)

3. In order to make the robot go forward, you'll want both motor C *and* motor B to run forward. The command `motor[motorC]=100;` made Motor C move at 100% power. Add a command that is exactly the same, but addresses Motor B instead.

```

1  task main()
2  {
3
4      motor[motorC] = 100;
5      motor[motorB] = 100;
6      wait1Msec(3000);
7
8  }

```

3. Add this code

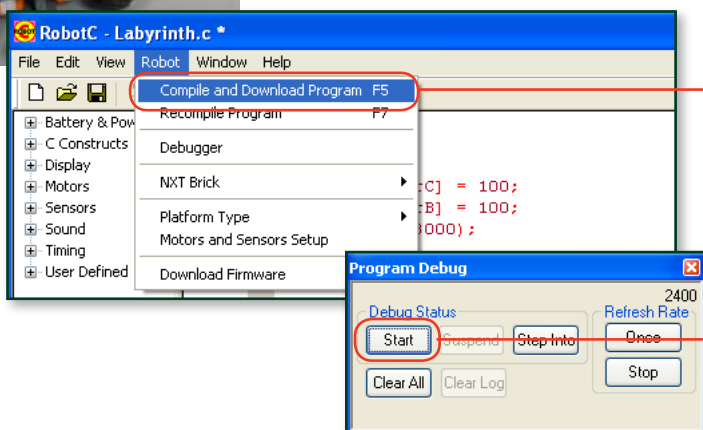
This code is exactly the same as the line above it, except that it is directed at Motor B (right wheel) instead of Motor C (left wheel).

4. Make sure your robot is on and that the robot is plugged in with the USB cable, then go to the menu "Robot" > "Compile and Download".



4a. Check connection

Ensure that your robot is turned on and plugged in to the computer through the USB cable



4b. Compile and Download

Select Robot > Compile and Download Program to send your program to the robot.

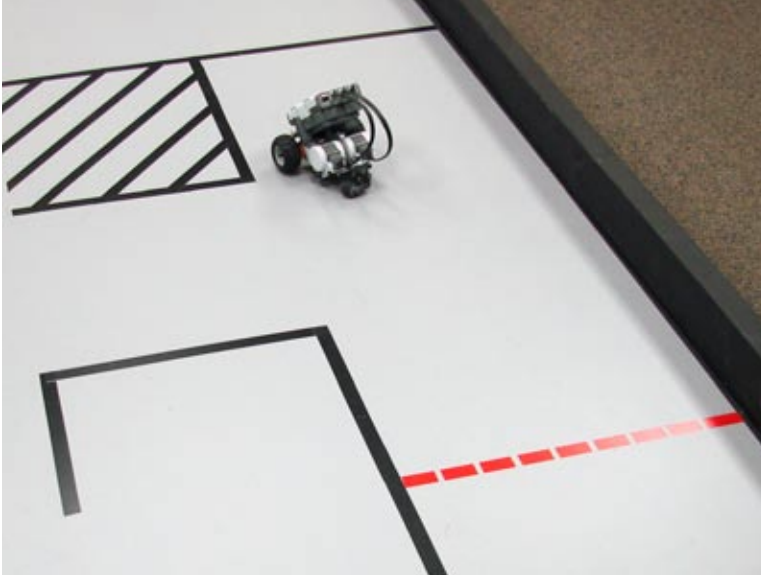
4c. Press Start

Press the Start button on the Program Debug menu that appears, to run the program.

Movement

Moving Forward **Code Dissection** (cont.)

5. Once the program is downloaded, you can either unplug the bot and navigate to your program to run it, or you can keep it connected to the computer and click on the "Start" button.



End of Section

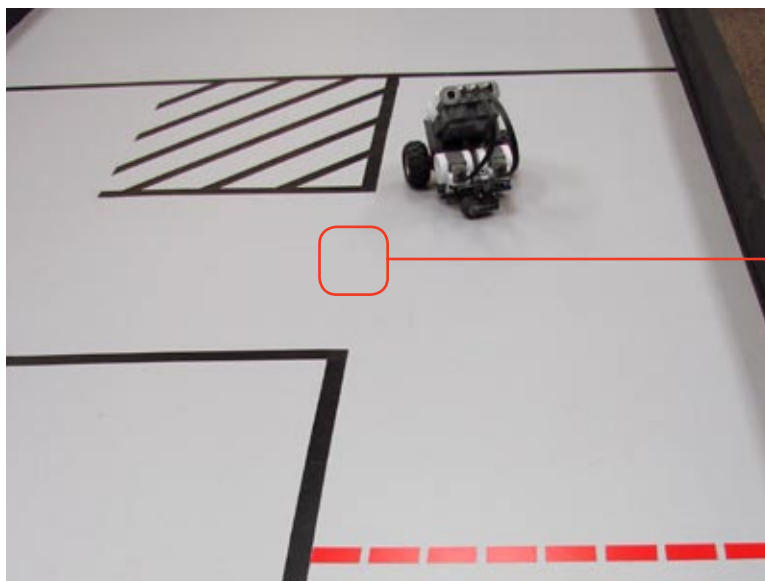
By examining what each line of code in the Sample Program did, we were able to figure out a way to turn on the other motor on the robot as well. Both motors running together created a forward movement. Proceed to the next section to begin experimenting with the other parts of the program.

Movement

Moving Forward Timing Lesson

In this lesson, you will learn how to adjust the time (and consequently, the distance) the robot travels in the Moving Forward behavior.

The robot moves forward for 3 seconds. This is a great start, but the end needs work.



Missed turn

The robot has traveled too far and cannot make the first turn in the maze.

1. Adjust the amount of time the robot lets its motors run, by changing the number value inside the wait1Msec command.

```

1 task main()
2 {
3
4     motor[motorC] = 100;
5     motor[motorB] = 100;
6     wait1Msec(2000);
7
8 }
```

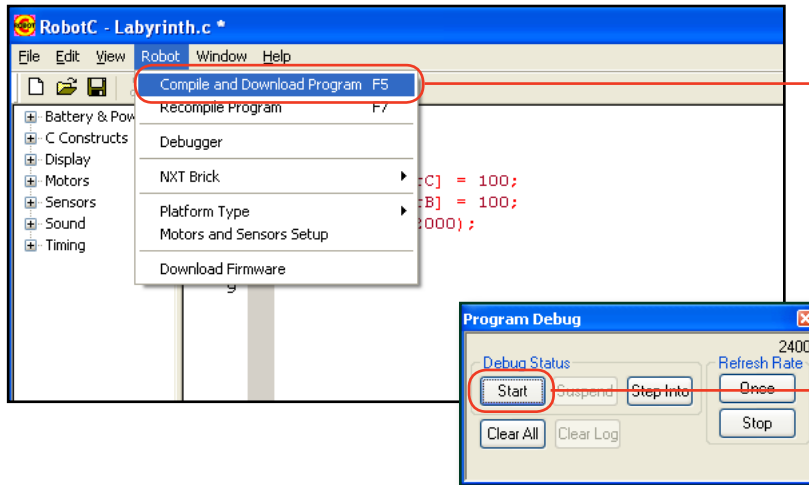
1. Modify this code

Change the 3000 milliseconds in the wait1Msec command to 2000 milliseconds.

Movement

Moving Forward **Timing** (cont.)

2. Compile and Download the program by going to "Robot" > "Compile and Download".



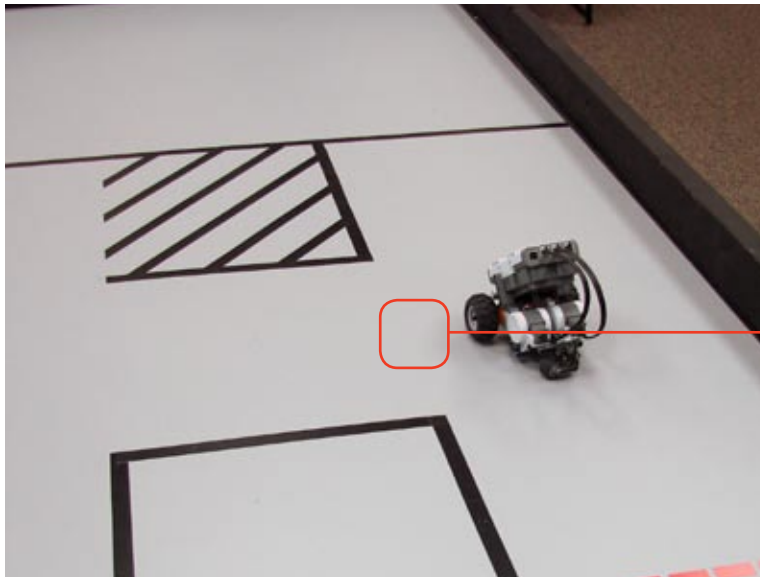
2a. Compile and Download

Select Robot > Compile and Download Program to send your program to the robot.

2b. Press Start

Press the Start button on the Program Debug menu that appears, to run the program.

End of Section. The wait1Msec command controlled how long the robot let its motors run. By shortening the duration from 3000ms to 2000ms, we adjusted the total distance traveled as well.



Ready to turn

The robot stops in a good position to begin its next maneuver, a left turn toward the next part of the path.