CHAPTER 18

Title = Anything You Like

That's a strange title, isn't it? You can pick anything you like and make it the title. But that doesn't mean the material covered in this chapter will change. Nope, this chapter covers a special type of block called the `VARIABLE` block. You'll find this block useful when you need to store a piece of information for later use. So let's take a look.

The VARIABLE Block

Let's imagine for just a moment that you want to give SPOT some information to remember. This information consists of some words, a few numbers, and a couple of logical True/False values. SPOT has the ability to place each piece of information in a virtual folder that exists in his memory. Here's the pseudo-code:

```
Me: SPOT, will you please store the words "pizza" and "cheesecake" in your memory?
Me: SPOT, I also need you to store the numbers "50" and "200" in your memory.
Me: SPOT, will you also please store one logical "True" and one logical "False" in your
memory?
```

Now, before we convert this pseudo-code to an NXT-G program, I need to tell you a little bit about how an NXT-G program stores information, changes it, and retrieves it. All of this is done using the `VARIABLE` block.

A `VARIABLE` block can do one of two things:

- Information can be *written to* a virtual folder that is stored in memory.
- Information can be *read from* a virtual folder that is stored in memory.

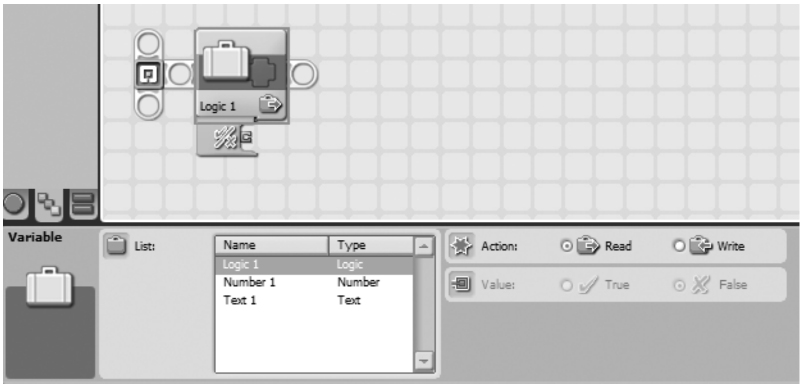
These virtual folders are also known as *variables*. An NXT-G variable can only be configured to hold one of three types of data: Text, Number, or Logic (True or False).

Text is easy enough; my pseudo-code tells SPOT to store "pizza" and "cheesecake," but it could just as easily have told SPOT to store the letter "A" or the sentence "My name is SPOT."

Numbers are even easier: when an NXT-G `VARIABLE` block is configured to hold a number, it can *only* be a positive or negative integer. Numbers such as 4.5 or –10.2 will be rounded to the nearest integer (5 and –10 for my examples).

Logic values only have two choices: True or False. An NXT-G `VARIABLE` block configured to hold a logical value can hold *only* True or False and nothing else.

OK, now it's time to take a look at the VARIABLE block. This block is found on the Complete Palette on the Data fly-out menu. Select the VARIABLE block, and drop it on the beam (see Figure 18-1).



List:

Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text

Action: [✓ Read] Write

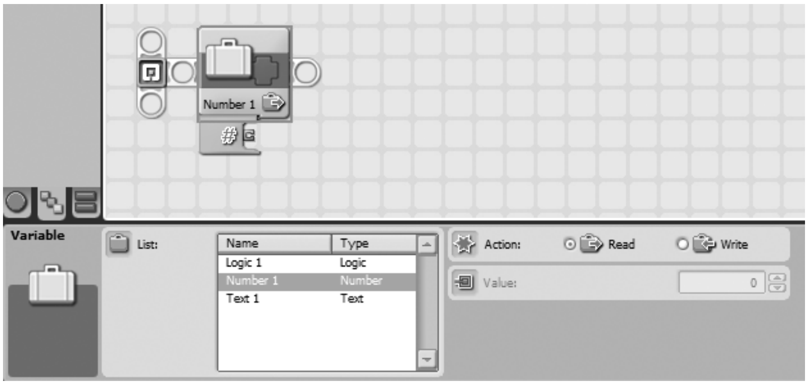
Value: True [✓ False]

Figure 18-1. *The VARIABLE block and its configuration panel*

I mentioned to you that information can be *read from* or *written to* a VARIABLE block. In Figure 18-1, you'll notice that, by default, the first time you drop a block on the beam it is configured to Read (in the Action section) a Logic Type value. The variable also has a Name assigned to it: Logic 1.

This means that if True or False is stored in the variable, this value can be *read from* the variable. Notice that the Value section is grayed out; it isn't available for you to edit. Also notice that the default value selected in the Value section is False.

Before I show you how to change this, select the variable named Number 1 in the VARIABLE block's configuration panel (see Figure 18-2).



List:

Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text

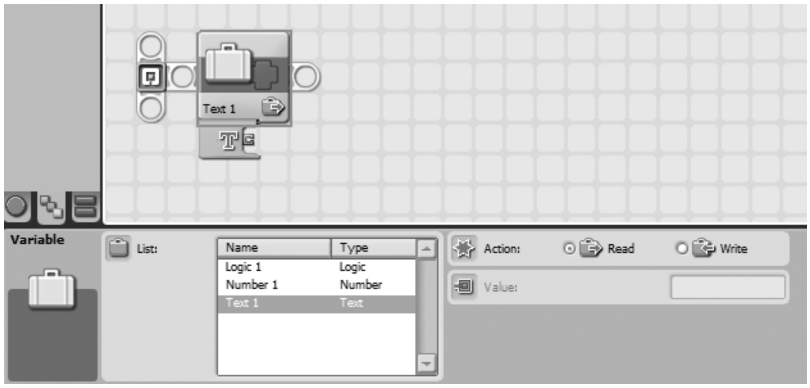
Action: [✓ Read] Write

Value: 0

Figure 18-2. *The VARIABLE block with Number 1 selected as the variable name*

If you choose Number 1 as the variable name, you'll see that the default value stored is zero (0). This number is the value that will be *read from* the variable named Number 1.

Next, choose Text 1 in the `VARIABLE` block's configuration panel (see Figure 18-3).



List:

Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text

Action: [✓ Read] Write

Value:

Figure 18-3. *The VARIABLE block with Text 1 selected as the variable name*

If you choose Text 1 as the variable name, the default value stored is blank; there is no text stored in the Text 1 variable.

In Figures 18-1, 18-2, and 18-3, notice also that the each of the blocks has only one output data plug. This matches what we know about a `VARIABLE` block with the Action section configured to Read. The variables can *only* be *read from*; other blocks (a `DISPLAY` block, for example) would use a data wire from this output data plug as input. This is "reading from" the variable.

I want to make sure you understand this concept, so take a look at Figure 18-4, and I'll explain it further.

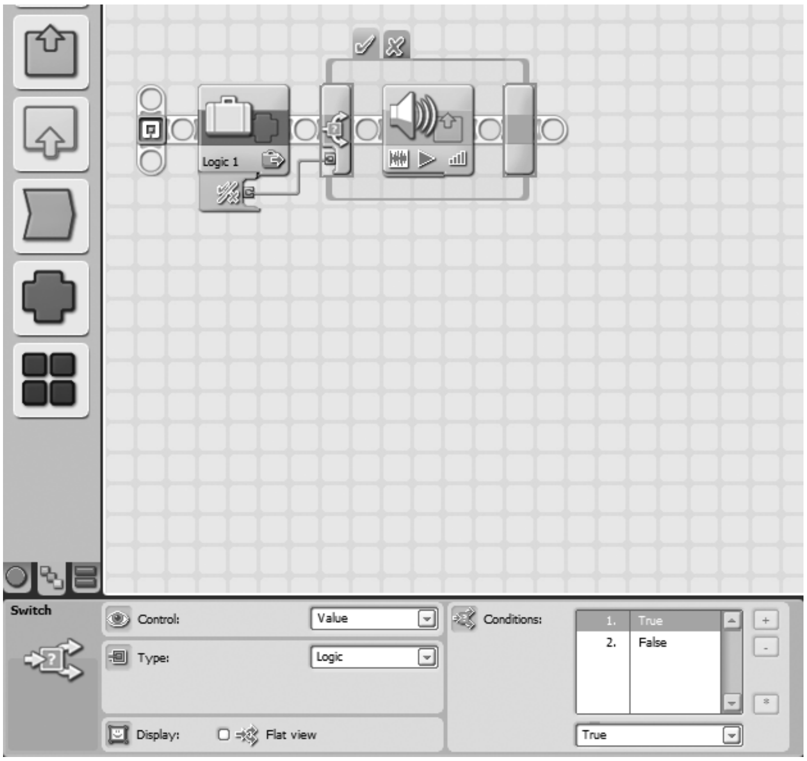
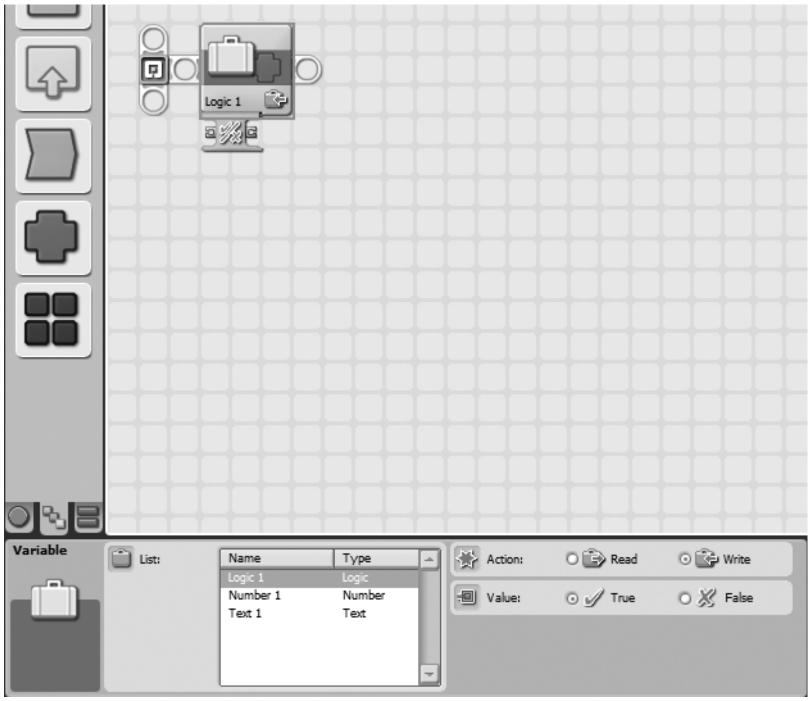


Figure 18-4. *A VARIABLE block providing its variable value to another block*

In Figure 18-4, I've connected a `VARIABLE` block called Logic 1 to a `SWITCH` block. The `SWITCH` block's configuration panel is visible in Figure 18-4, and you can see that I've configured it to check for a Logic value in the Type section. When the program is executed, the `SWITCH` block will *read* the value from the `VARIABLE` block. If the value is True, the `SWITCH` block will execute the `SOUND` block I've dropped in the True tab. If the value is False, the `SWITCH` block will execute a `MOVE` block that I dropped in the False tab.

Now, that example involved reading the data directly from the `VARIABLE` block. What if I want to put some data in? Then I'd have to *write* some data into the block. Here is how you do it.

Take a look at Figure 18-5. In this figure, I've selected Write in the Action section.



Variable

List:

Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text

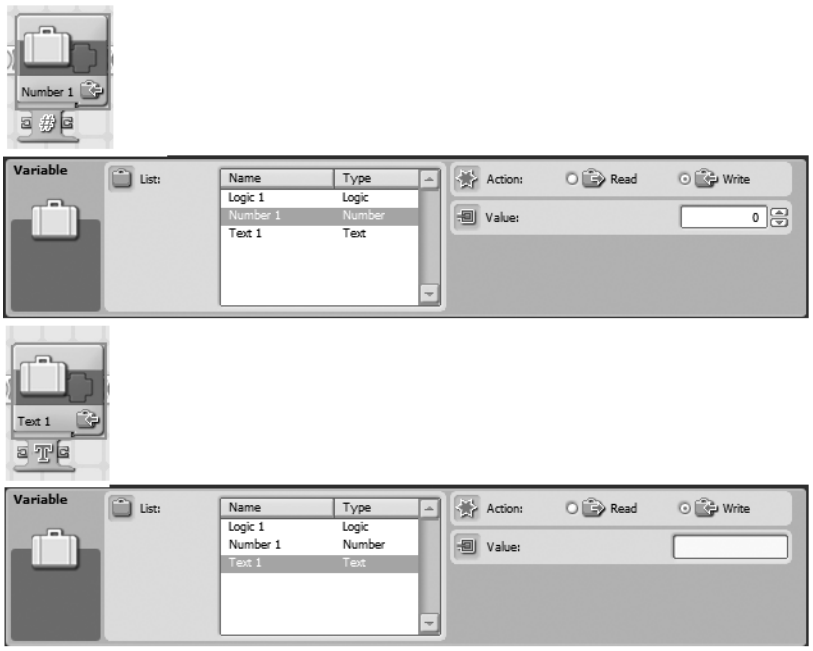
Action: Read [✓ Write]

Value: [✓ True] False

Figure 18-5. *The VARIABLE block with Write selected in the Action section*

In Figure 18-5, I can change the logic value. Remember that the default is False, so I've changed it to True. Did you also notice the input data plug that was added on the data hub? This means that I could actually have the True or False value determined by a data wire from another block. (In this example, the data wire *must* carry a logic value of True or False; anything else would give me a broken wire that wouldn't work. Likewise, if the List section has Number or Text selected, the input data wire *must* be providing a data type of the selected item.)

Figure 18-6 shows the configuration panels for the Number 1 and Text 1 blocks. I'm including them, because I want you to see how you can modify the data when the Action option is set to Write. For the Number 1 variable, you can type either a positive or a negative integer in the Value field. For the Text 1 variable, you can type words, sentences, or even entire paragraphs in the Value field.



Variable

List:

Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text

Action: Read [✓ Write]

Value: 0

Variable

List:

Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text

Action: Read [✓ Write]

Value:

Figure 18-6. *The VARIABLE block with Write selected for the Number 1 and Text 1 variables*

After you've modified the data in a variable block, you can either close the variable by selecting Read in the Action menu (this prevents the data from being changed) or leave the block alone. If you choose to leave the Action section option as Write, you can drag a data wire from another block to the input data plug on the variable block (just remember that the type of data wire going into the input data plug must match the type of data the variable is configured to hold—Text, Number, or Logic).

Now, let's go back to the original pseudo-code I gave SPOT:

Me: SPOT, will you please store the words "pizza" and "cheesecake" in your memory?

Me: SPOT, I also need you to store the numbers "50" and "200" in your memory.

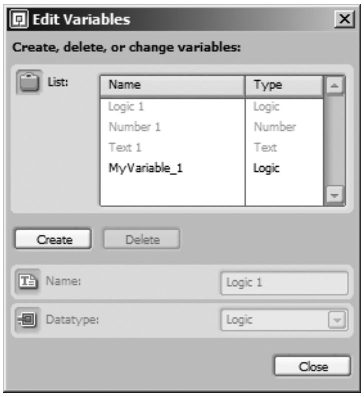
Me: SPOT, will you also please store one logical "True" and one logical "False" in your memory?

Do you see a problem? I've got two pieces of text to store and two number values. I already know that a variable block can only hold *one* piece of data. If I store the number 50, for example, in the Number 1 variable, where will I store the number 200? And I can store "pizza" in the Text 1 variable, but what about "cheesecake?"

The answer is simple. NXT-G allows you to create as many variables as you need. You can even rename a variable from something like Text 1 to something easier to remember like "Food", or Number 1 to something like "Test Scores". Here's how it's done.

If you need to create another variable, start by clicking the Edit menu at the top of the NXT-G software. Click the menu option labeled Define Variables.

A small window, like the one shown in Figure 18-7, opens.



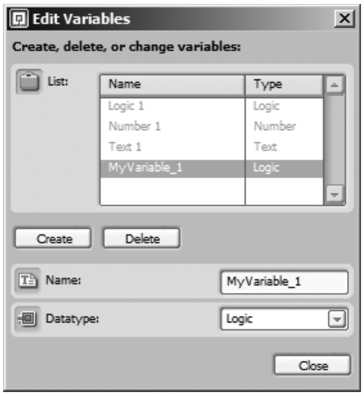
Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text
MyVariable_1	Logic

Name: Logic 1

Datatype: Logic

Figure 18-7. *The Edit Variables window allows you to create new variables.*

Click the Create button, and a new variable will appear in the List section with a default name like MyVariable_1 (see Figure 18-8).



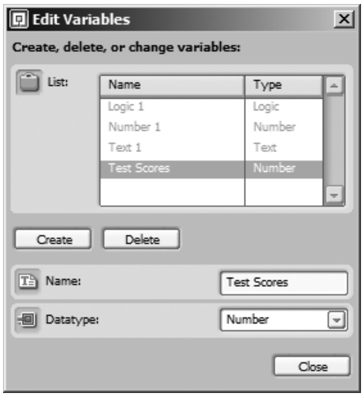
Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text
MyVariable_1	Logic

Name: MyVariable_1

Datatype: Logic

Figure 18-8. *Your new variable appears in the List section.*

You can change the name of the variable by typing a new name in the Name field. Select the type of data the variable will contain in the drop-down menu in the Datatype section. I've named my new variable Test Scores and configured it to hold a Number (see Figure 18-9). You can click the Create button again to make another variable. Click the Close button when you are finished.


Edit Variables

Create, delete, or change variables:

List:

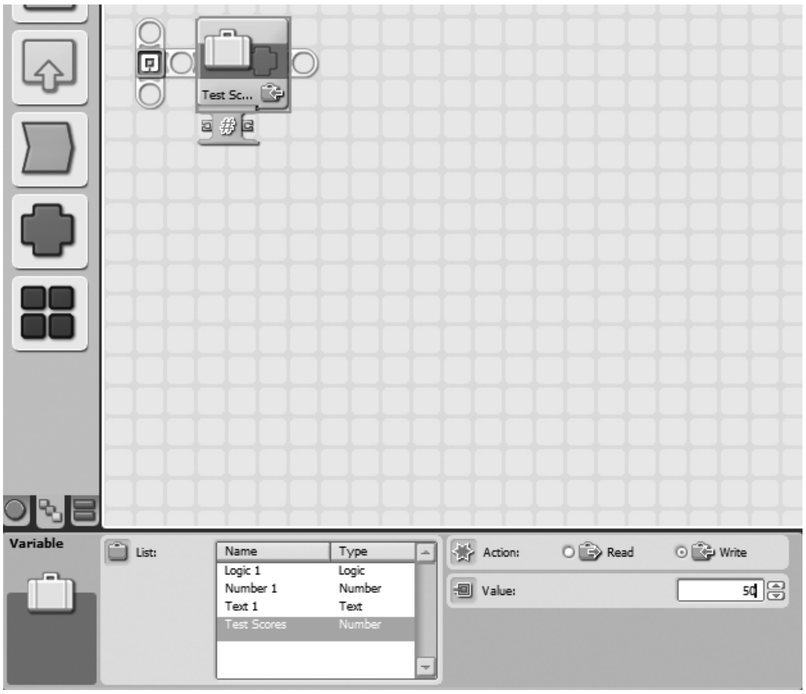
Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text
Test Scores	Number

Name: Test Scores

Datatype: Number

Figure 18-9. *Give the new variable a Name and a Datatype to hold.*

Now, when you drop a `VARIABLE` block on the beam, you'll notice that your new variable appears as a selection in the List section (see Figure 18-10). Using this method, you can create as many Text, Number, and Logic variables as you need.



List:

Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text
Test Scores	Number

Action: Read [✓ Write]

Value: 50

Figure 18-10. *Your new variable now appears in the List section of the configuration panel.*

OK, now you know how to put data into a variable (Write) and how to get data from a variable (Read). You also know how to create additional variables. The last thing I want to cover in this chapter is how to use variables throughout your programs.

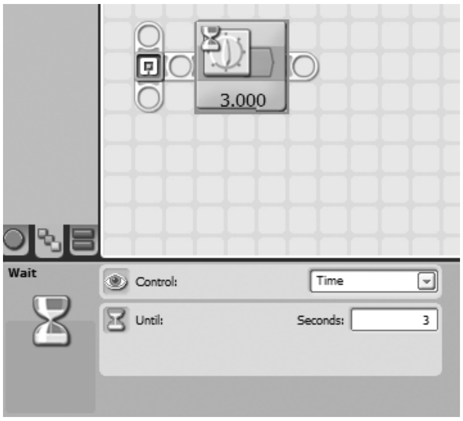
Here's a bit of pseudo-code for SPOT:

Me: SPOT, I want you to wait for me to press and release the Enter button (orange button) for 3 seconds.

Me: If I press and release the button, display "Pressed" on the screen.

Me: If I do not press and release the button, display "Not Pressed" on the screen.

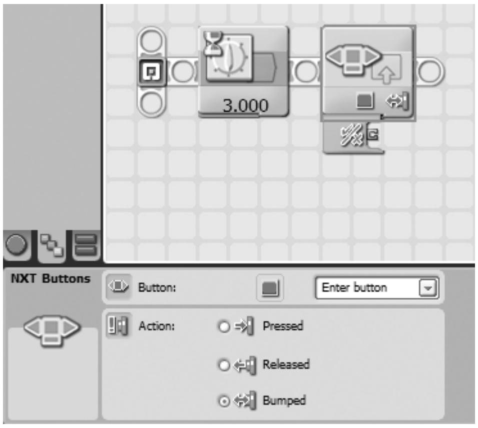
I'll first start off with a `WAIT` block configured to loop for 3 seconds (see Figure 18-11).



Control: Time
Until: Seconds: 3

Figure 18-11. *The WAIT block is configured to loop for 3 seconds.*

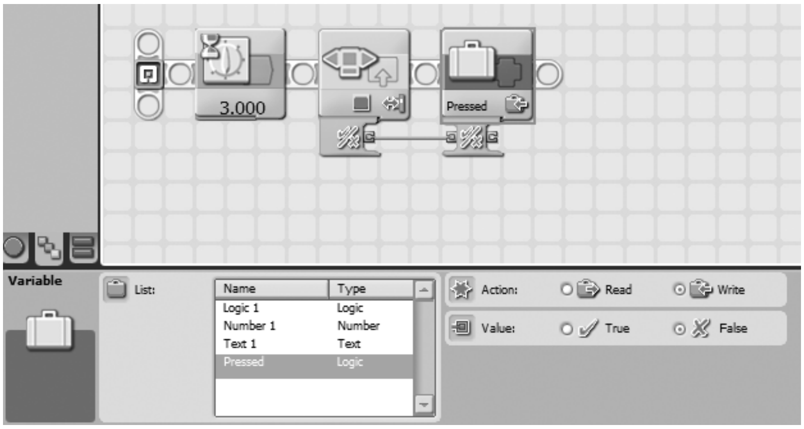
I'll next drop in an `NXT BUTTONS SENSOR` block and configure it to detect the press and release (Bumped) of the Enter button (see Figure 18-12).



Button: Enter button
Action: Pressed / Released / [✓ Bumped]

Figure 18-12. *The NXT BUTTONS SENSOR block will monitor the Enter button.*

I created a new `VARIABLE` block called "Pressed" that holds a Logic value. I next drop a `VARIABLE` block after the `NXT BUTTONS SENSOR` block. I choose my new variable, Pressed, and in the Action section, I choose Write. I also drag a data wire out of the `NXT BUTTONS SENSOR` block's output data plug into the input data plug on the `VARIABLE` block (see Figure 18-13).



Variable

List:

Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text
Pressed	Logic

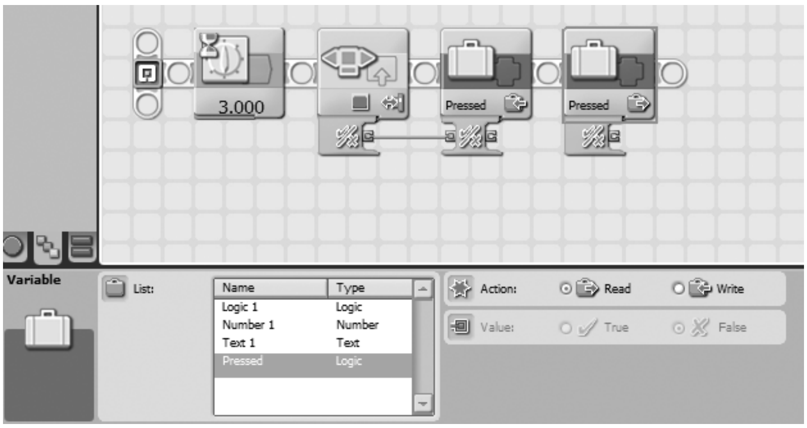
Action: Read [✓ Write]

Value: True [✓ False]

Figure 18-13. *The variable Pressed will hold True or False.*

If the Enter button is pressed and released during the first 3 seconds, the `NXT BUTTONS SENSOR` block will detect the press and release and change the logic value to True. A value of True or False will continue to be written to the Pressed variable block until the 3 seconds expire. When the program starts, the initial value will be False, but it can change anytime to True if the Enter button is bumped.

When the `WAIT` block ends (after 3 seconds), we need to display "Pressed" or "Not Pressed" on the LCD screen. To do this, we'll drop in another `VARIABLE` block. This time I'm going to choose my Pressed variable again, but I'm changing its Action section option to Read (see Figure 18-14).



Variable

List:

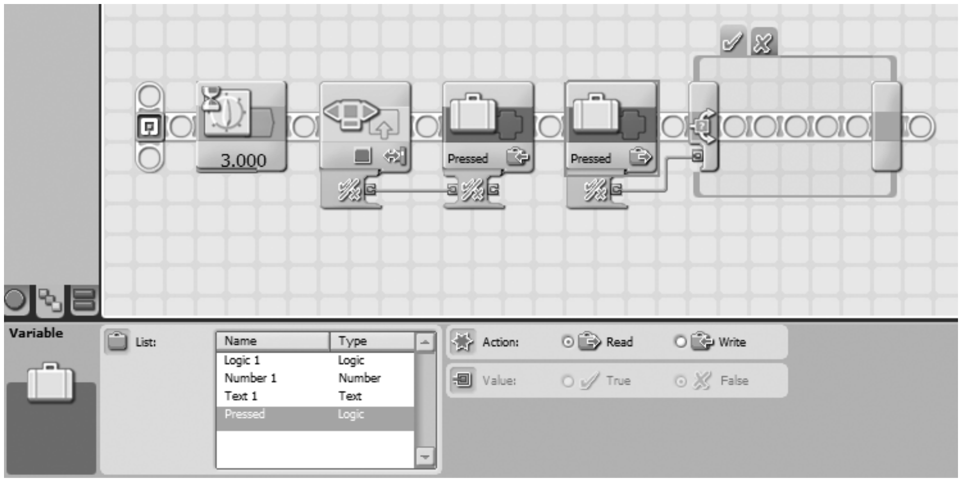
Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text
Pressed	Logic

Action: [✓ Read] Write

Value: True [✓ False]

Figure 18-14. *The variable Pressed will be read from by a SWITCH block.*

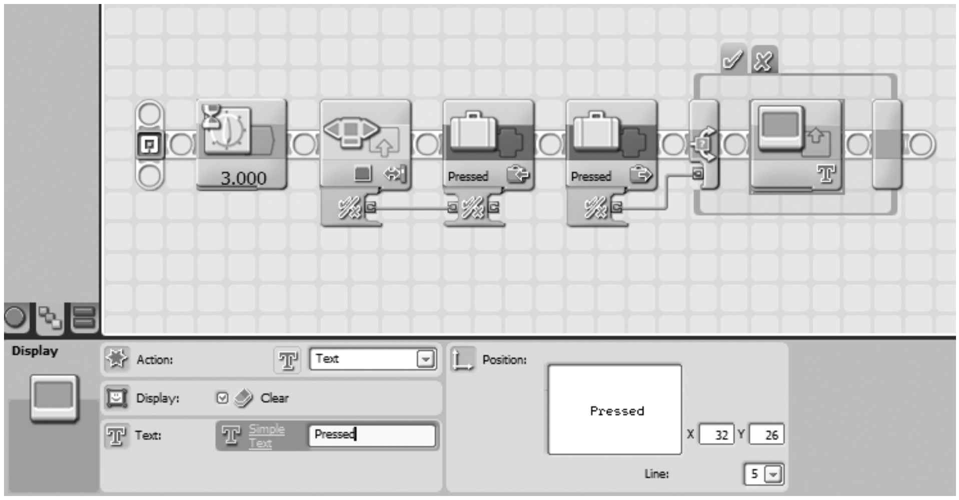
Now, I drop in a SWITCH block and configure it to read a Logic value (see Figure 18-15), and I've turned off Flat view so the SWITCH block is visible in its tabbed format. I've also dragged a data wire from the VARIABLE block to the SWITCH block that contains the value True or False.



Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text
Pressed	Logic

Figure 18-15. *The SWITCH block will read the Pressed variable's value.*

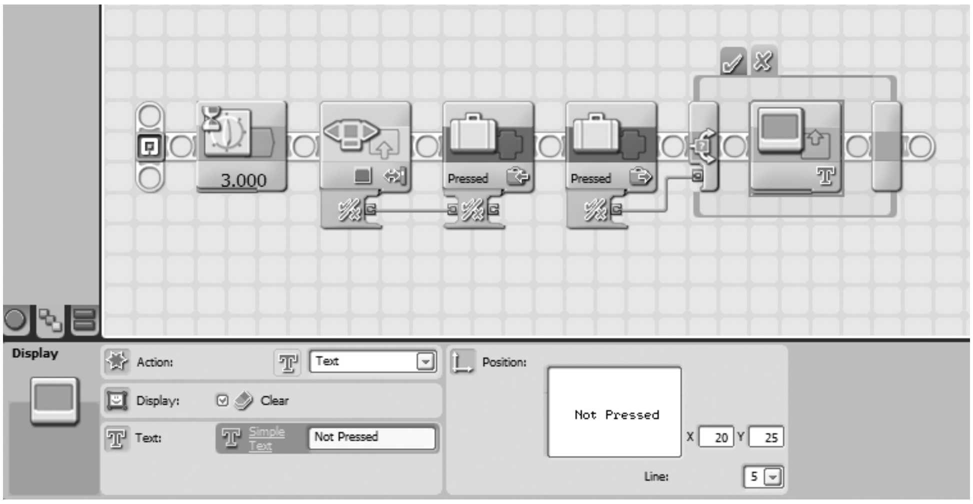
In the True tab, I place a DISPLAY block that clears the LCD screen and puts the word "Pressed" on the screen (see Figure 18-16).



Action: Text
Display: ☑ Clear
Text: Pressed
Pressed
X 32 Y 26
Line: 5

Figure 18-16. *This DISPLAY block will display "Pressed" if the variable has a value of True.*

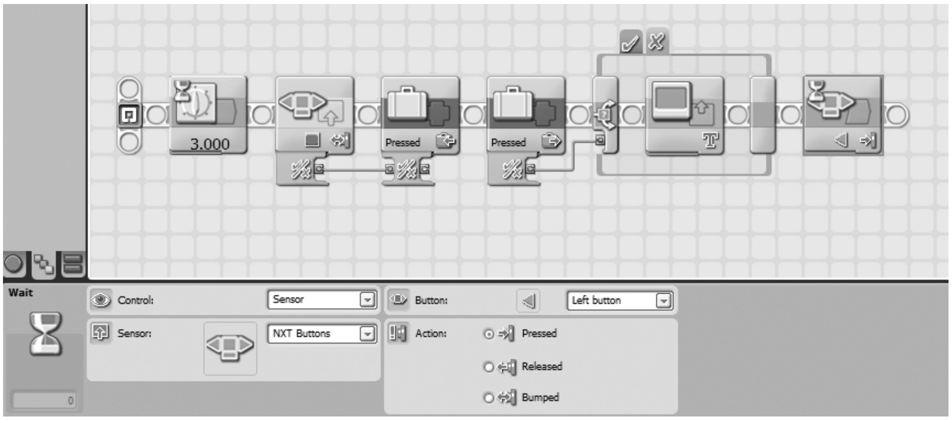
On the False tab, I place another DISPLAY block that clears the LCD screen and puts the words "Not Pressed" on the screen (see Figure 18-17).



Action: Text
Display: ☑ Clear
Text: Simple Text Not Pressed
Position:
Not Pressed
X 20 Y 25
Line: 5

Figure 18-17. *The words "Not Pressed" will display if the variable has a value of False.*

Finally, I drop in an NXT BUTTON WAIT block to wait until the left button is pressed; this will give me time to view the results on the screen (see Figure 18-18).



Control: Sensor
Sensor: NXT Buttons
Button: Left button
Action: [✓ Pressed] Released Bumped

Figure 18-18. *If I press the left button, the program will end.*

Now run the program. Try it a few times—press and release the Enter button, or don't press and release it. Your decision to bump or not bump the button will be converted to a True or False value that is written to the Pressed variable. After 3 seconds, the Pressed variable is read by the SWITCH block, and the proper text is written on the LCD screen.

Variables are a powerful way for your robot to store away information—and to use that data later. Once the variable has been created and data written to it, that data will be available anytime you need it—well, at least until the program ends. If you want to store data for use after the program ends and/or after the power has been turned off, you'll need a different type of block; we'll discuss that later in Chapter 22. But in the next chapter, I'll show you how to use the TEXT block.