Variables

Variables are places to store values (such as sensor readings) for later use, or for use in calculations. There are three main steps involved in using a variable:

1. Introduce (create or "**declare**") the variable
2. Give ("**assign**") the variable a value
3. Use the variable to access the stored value

```
task main()
{
    int speed;

    speed = 75;

    motor[motorC] = speed;
    motor[motorB] = speed;
    wait1Msec(2000);
}
```

Declaration
The variable is created by announcing its **type**, followed by its **name**. Here, it is a variable named **speed** that will store an integer.

Assignment
The variable is assigned a value. The variable **speed** now contains the integer value **75**.

Use
The variable can now "stand in" for any value of the appropriate type, and will act as if its stored value were in its place.

Here, both motor commands expect integers for power settings, so the int variable **speed** can stand in. The commands set their respective motor powers to the **value** stored in **speed**, 75.

In the example above, the variable "speed" is used to **store a number**, and then retrieve and **use that value** when it is called for later on. Specifically, it stores a number given by the programmer, and retrieves it twice in the two different places that it is used, once for each of the motor commands. This way both motors are set to the same value, but more interestingly, you would only need to **change one line of code to change both motor powers**.

```
task main()
{
    int speed;

    speed = 50;

    motor[motorC] = speed;
    motor[motorB] = speed;
    wait1Msec(2000);
}
```

One line changed
The value assigned to **speed** is now 50 instead of 75.

Changed without being changed
No change was made to the program here, but because these lines use the value contained in the variable, both lines now tell their motors to run at a power level of 50 instead of 75.

This example shows **just one way** in which variables can be used, as a convenience for the programmer. With a robot, however, the ability to store sensor values (values that are **measured by the robot, rather than set by the programmer**) adds invaluable new capabilities. It gives the robot the ability to take measurements in one place and deliver them in another, or even do its own calculations using stored values. The same basic rules are followed, but the possibilities go far beyond just what you've seen so far!

Variables

Declaration Rules
In order to declare a variable, you must declare its type, followed by its name. Here are some specifics about the rules governing each:

Rules for Variable Types
• You must choose a data type that is appropriate for the value you want to store

The following is a list of data types most commonly used in ROBOTC:

Data Type	Description	Example	Code
Integer	Positive and negative whole numbers, as well as zero.	-35, -1, 0, 33, 100, 345	`int`
Floating Point Decimal	Numeric values with decimal points (even if the decimal part is zero).	-.123, 0.56, 3.0, 1000.07	`float`
String	A string of characters that can include numbers, letters, or typed symbols.	"Counter reached 4", "STOP", "time to eat!"	`string`
Boolean	True or False. Useful for expressing the outcomes of comparisons.	true, false	`bool`

Rules for Variable Names

• A variable name can not have **spaces** in it

• A variable name can not have **symbols** in it

• A variable name can not **start with a number**

• A variable name can not be the same as an existing **reserved word**

Proper Variable Names	Improper Variable Names
linecounter	line counter
threshold	threshold!
distance3	3distance
timecounter	time1[T1]

Variables

Assignment and Usage Rules
Assignment of values to variables is pretty straightforward, as is the use of a variable in a command where you wish its value to be used.

Rules for Assignment

• Values are assigned using the **assignment operator** = (not ==)

• Assigning a value to a variable that already has a value in it will **overwrite** the old value with the new one

• **Math operators** (+, -, *, /) can be used with assignment statements to perform calculations on the values before storing them

• A variable can appear in **both the left and right hand sides** of an assignment statement; this simply means that its **current value** will be used in calculating the new value

• Assignment can be done in the same line that a variable is **declared** (e.g. `int x = 0;` will both create the variable x and put an initial value of 0 in it)

Rules for Variable Usage

• "Use" a variable simply by putting its name where you want its value to be used

• The current value of the variable will be used every time the variable appears

Examples:

Statement	Description
`motorPower = 75;`	Stores the value 75 in the variable "motorPower"
`lightVariable = SensorValue(lightSensor);`	Stores the current sensor reading of the sensor "lightSensor" in the variable "lightVariable"
`sum = variable1 + variable2;`	Adds the value in "variable1" to the value in "variable2", and stores the result in the variable "sum"
`average = (variable1 + variable2)/2;`	Adds the value in "variable1" and the value in "variable2", then divides the result by 2, and stores the final resulting value in "average"
`count = count + 1;`	Adds 1 to the current value of "count" and places the result back into "count" (effectively, increases the value in "count" by 1)
`int zero = 0;`	Creates the variable x with an initial value of 0 (combination declaration and assignment statement)