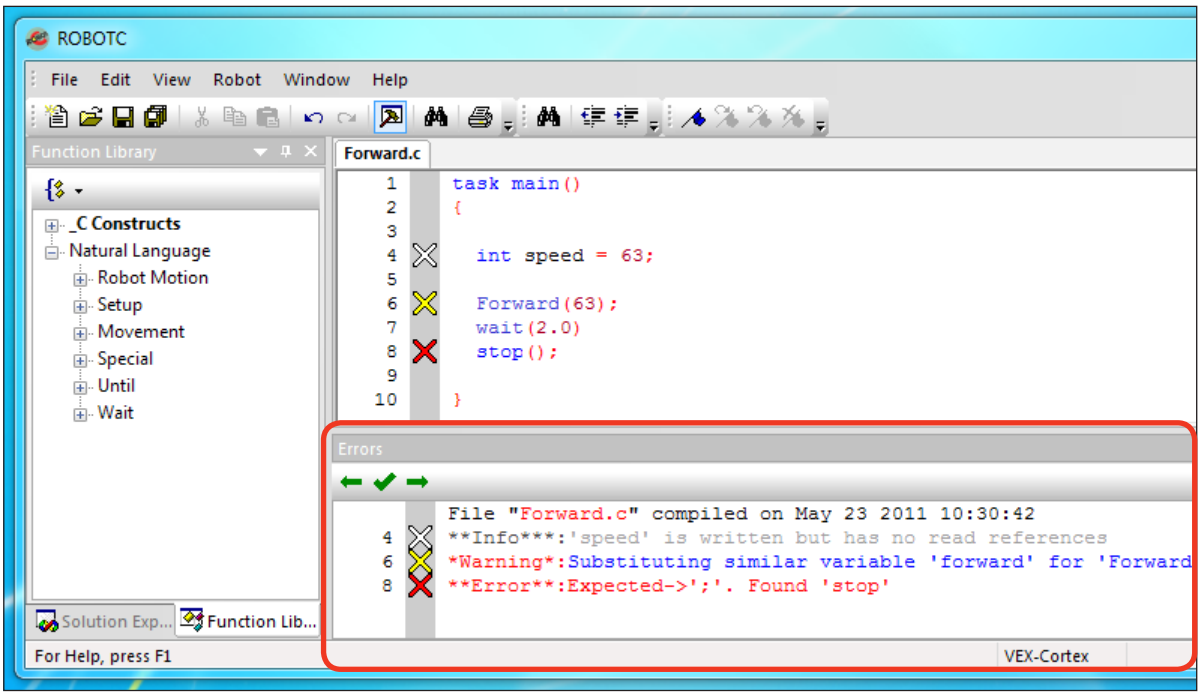
Error Messages in ROBOTC Code

ROBOTC has a built-in **compiler** that analyzes your programs to identify syntax errors, capitalization and spelling mistakes, and code inefficiency (such as unused variables). The compiler runs every time you download code to the robot and when you choose to compile your program from the **Robot** menu in ROBOTC.

Notifications regarding any errors, warnings and important information the compiler finds are displayed in the **Errors** display screen of the ROBOTC interface.



The Errors display screen reports the number of errors in your code, as well as their types. Double-clicking a compiler message in the Error display screen will highlight the relevant line of code in your program. Depending on the type of error, ROBOTC will only be able to highlight the approximate location. For instance, in the example above the missing semicolon is on line 7 but ROBOTC will highlight line 8.

ROBOTC generates **three types** of compiler messages: **Errors, Warnings and Information:**

Errors:
There was an issue your program that prevented it from compiling. These are usually misspelled words, missing semicolons, and improper syntax. Errors are denoted with a **Red X**.

Warnings:
There was a minor issue with your program, but the compiler was able to fix or ignore it. These are usually incorrect capitalizations or empty, infinite loops. Warnings are denoted with a **Yellow X**.

Information:
ROBOTC will generate information messages when it thinks you have declared functions or variables that are not used in your program. These messages inform you about inefficient programming. Information messages are denoted with a **White X**.

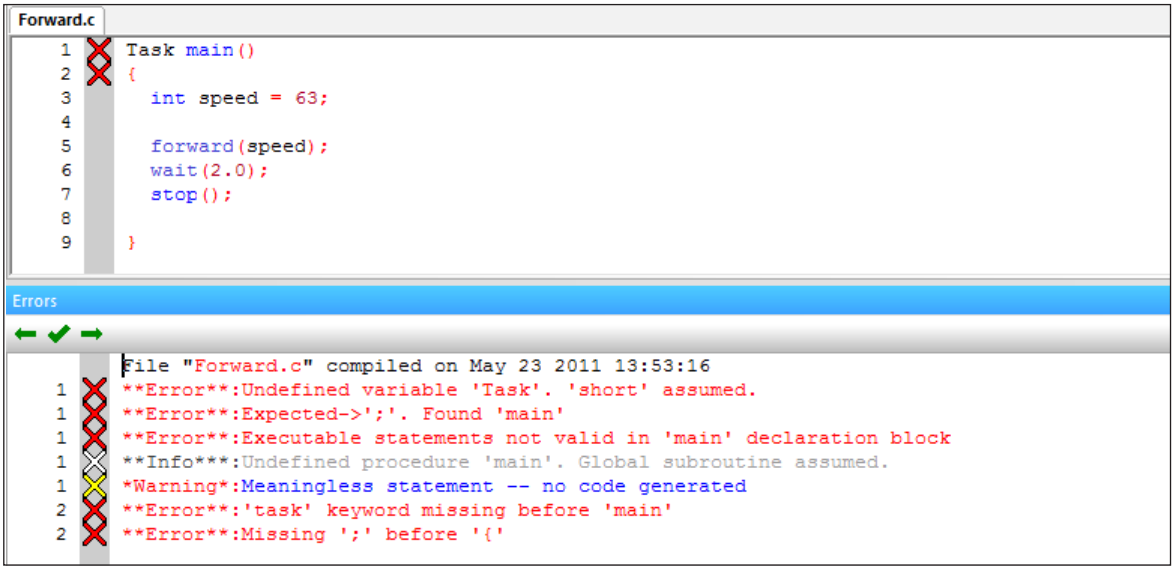
Error Messages in ROBOTC Code

Common Error Messages

Error messages will prevent your program from compiling and downloading to your robot. You must correct any and all error messages in your program before you will be able to download it to your robot. Also, **error messages can have a "ripple" effect**; errors at the beginning of your program can cause subsequent errors in the code. Because of this, it's recommended that you correct errors at the beginning of your program first, recompile your code, and then correct any remaining errors.

Most error messages are caused by misspelled reserved words and improper syntax. Many of these mistakes can be avoided by dragging commands from the Function Library into your ROBOTC programs.

In the example below, the T in task main is capitalized, causing multiple subsequent errors and warnings to appear in the Errors display screen.



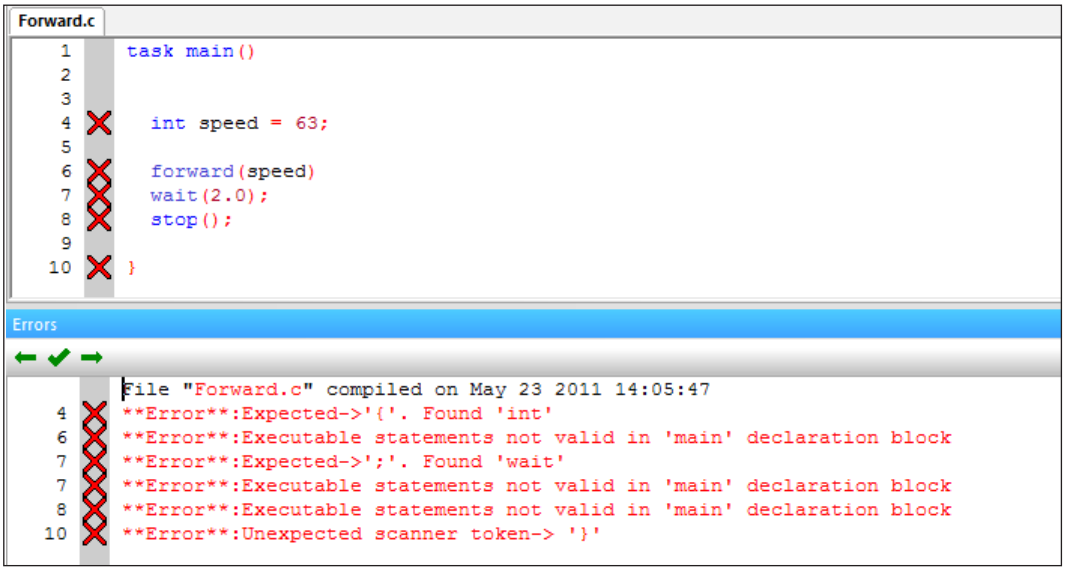
In situations like these, it's recommended that you try to correct the first error in your program before moving on. The first error message reads "***Error**:Undefined variable 'Task'. 'short' assumed.*" When the words "*Undefined variable*" appear in the Errors display screen, it indicates that ROBOTC does not recognize the specified word; the fact that Task is colored black instead of blue like other ROBOTC reserved words also indicates that ROBOTC does not recognize it.

To correct this error, you should replace the uppercase T with a lowercase t in task, and recompile your code. The compiler will reevaluate your code and generate a new set of notifications.

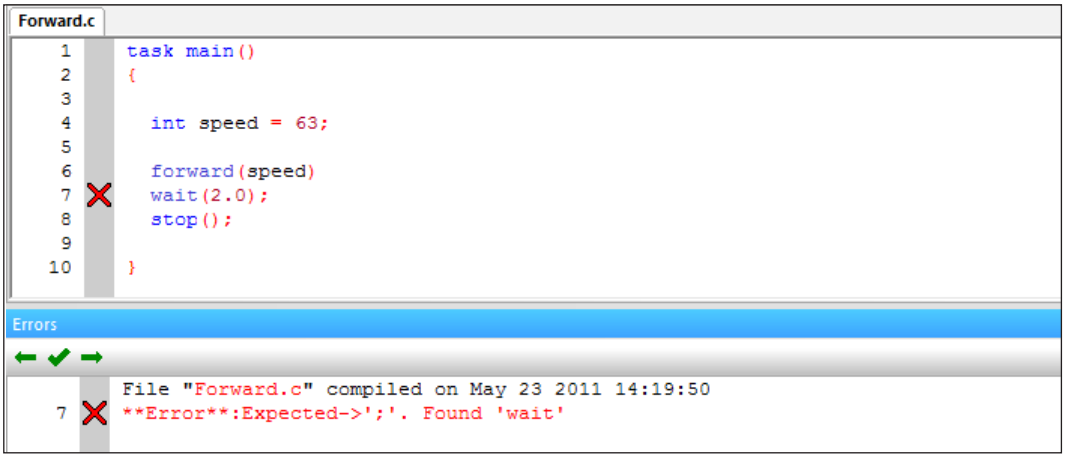
Error Messages in ROBOTC Code

Common Error Messages

The example below contains two syntax errors: a missing curly brace on line 2 and a missing semicolon on line 6. Once again, you should try to correct the first error in the program before moving on.



The first error message comes up on line 4, saying "***Error**:Expected->'{'. Found 'int'*". When the word "*Expected->*" appears in the Errors display screen, it usually indicates that a piece of syntax is missing. In this case, it expected to find the missing curly brace immediately after *task main()*, but found the reserved word *int* instead. To correct this error, you should add the opening curly brace on line 2 and then recompile your code.

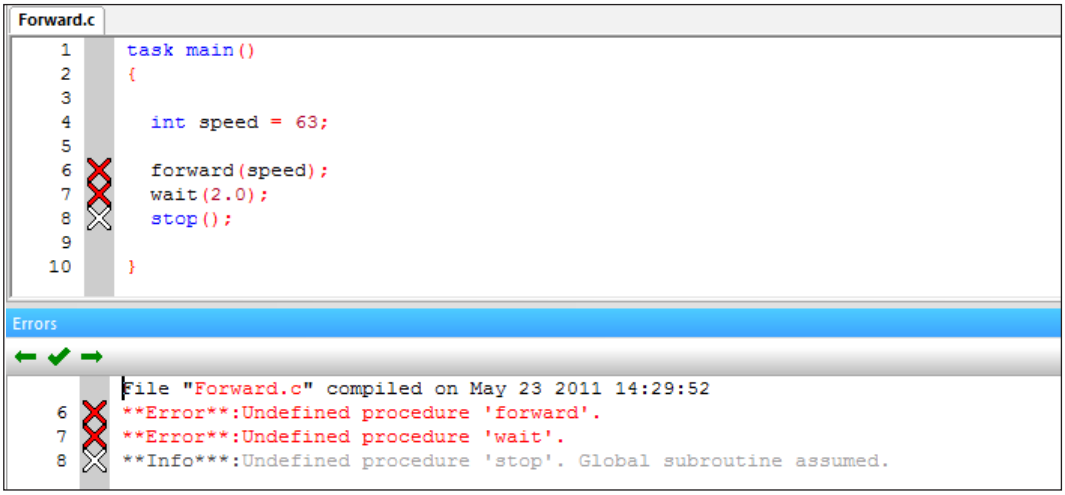


After recompiling your code, any remaining errors will be displayed. Missing semicolons also display an "*Expected->*" style error message, but notice that the error for the missing semicolon on line 6 appears on line 7. This is because the compiler ignores whitespace (blank lines, spaces and tabs), but expected a semicolon before it encountered the wait command. To correct this error, you should add a semicolon after the forward command.

Error Messages in ROBOTC Code

Common Error Messages

The example below, the ROBOTC compiler does not recognize the forward, wait or stop commands. Error messages that begin with "***Error**:Undefined procedure*" indicate that ROBOTC does not recognize the command; this is also indicated by the commands failing to turn blue like other ROBOTC reserved words.

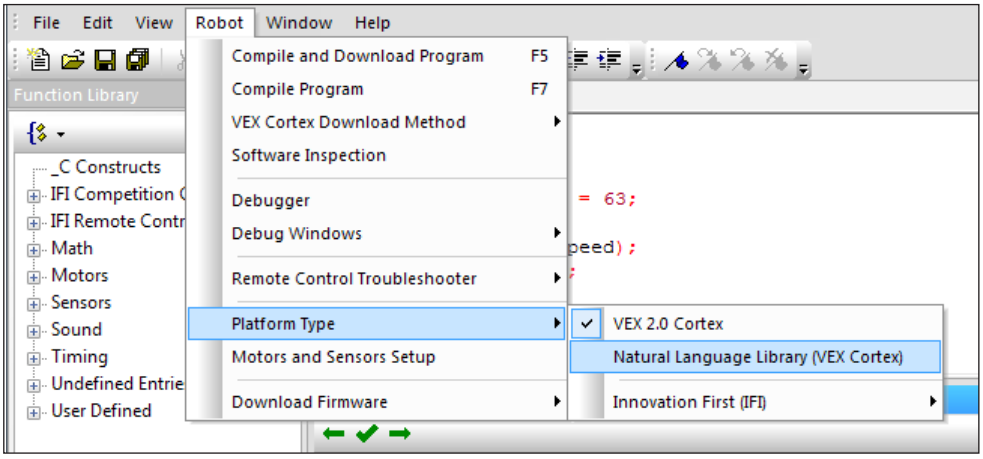


There are two main causes for this error:

1. The command is misspelled
2. The correct ROBOTC Platform Type is not selected.

To correct this error:

1. Verify that your spelling and capitalization is correct.
2. Verify that you have the appropriate Platform Type selected. If you are using Natural Language commands, you must have the Natural Language Platform Type selected.



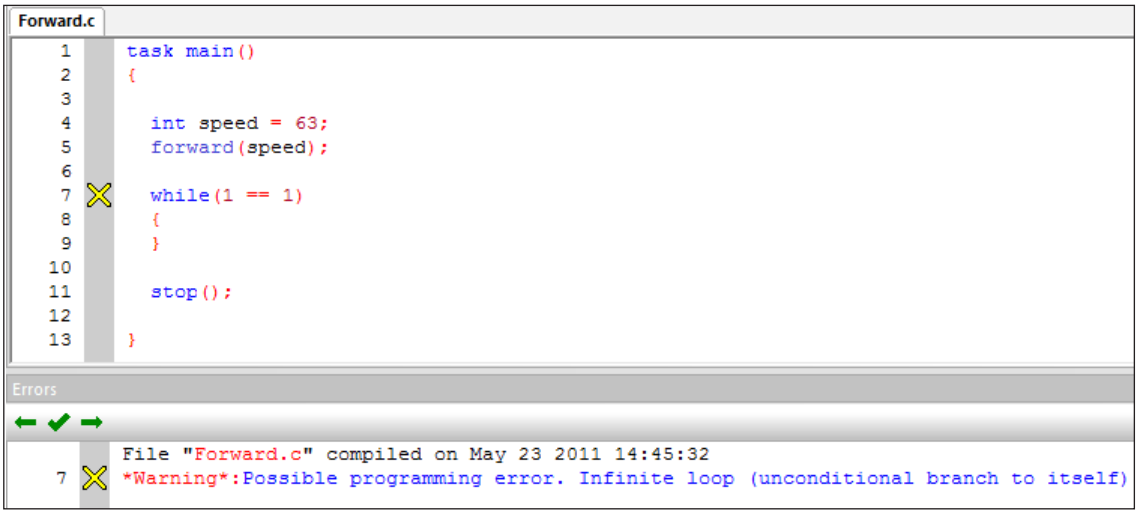
Error Messages in ROBOTC Code

Common Warning Messages

Warning messages are used to notify you about possible programming and logic errors in your program. With warning messages, the compiler is able to fix or ignore the issues so they will not prevent your program from compiling or downloading to your robot.

A common occurrence of warning messages are empty, infinite loops in your code. In the example below, the an infinite loop is created, with no code embedded within the loop to stop it from repeating forever. This is considered a warning, rather than an error, because it is valid code and can be intentionally used by a programmer.

The warning message will inform you that there was a possible programming error caused by an infinite loop: "**Warning*:Possible programming error. Infinite loop (unconditional branch to itself) detected.*"



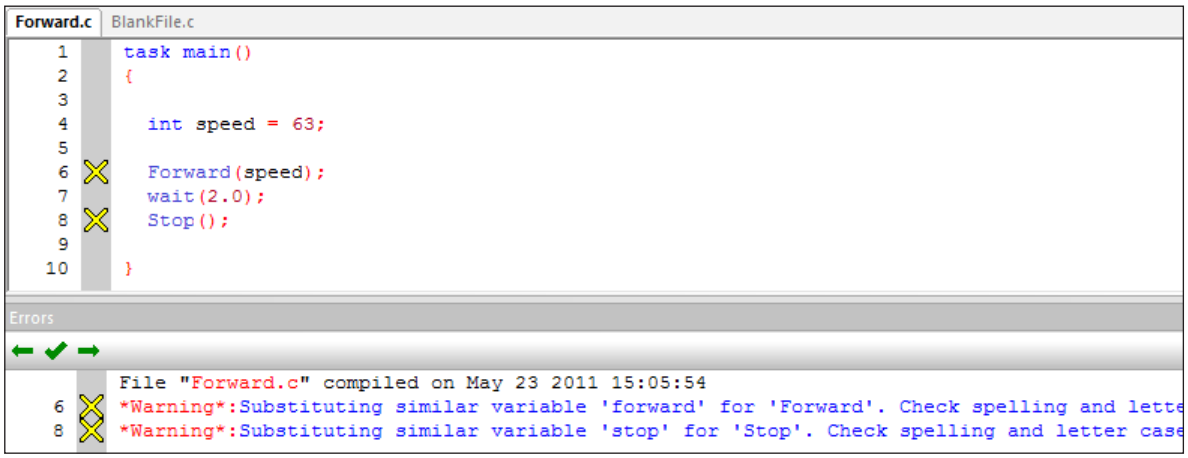
If creating an infinite loop was your intention, you **do not need** to correct this message. If it was not intentional, you can:

1. Include code within the curly braces of the while loop, for it to repeat.
2. Change the condition of the while loop, so that it does not repeat forever.

Error Messages in ROBOTC Code

Common Warning Messages

In the example below, the forward and stop commands are improperly capitalized. The ROBOTC compiler is able to substitute in the correct forms of the commands, but uses warning messages to notify you of the substitution. Note that it does **not** correct the capitalization in your code, only what it sends to the robot. Again, the actual warning message inform you about the substitution: *"*Warning*:Substituting similar variable 'forward' for 'Forward'. Check spelling and letter case."*



You **do not need** to correct this message. If it was not intentional, you can:

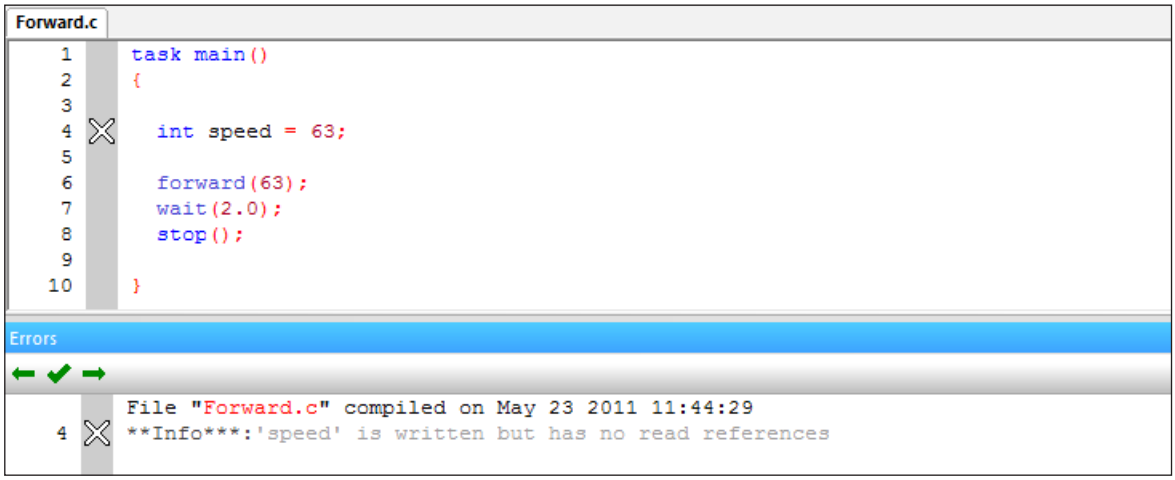
1. Include code within the curly braces of the while loop, for it to repeat.
2. Change the condition of the while loop, so that it does not repeat forever.

Error Messages in ROBOTC Code

Common Information Messages

Information messages will not prevent your program from compiling or downloading to your robot. They only notify you regarding possible inefficiencies in your code.

The most common occurrence of information messages are unused variables in your code. In the example below, the integer variable *speed* is created and initialized, but never actually used in the program. This is indicated in the Errors display screen with the message: "*'speed' is written but has no read references*".



You **do not need** to correct this message, but you can in two ways:

1. Eliminate the code on line number 4, deleting the variable.
2. Call the speed variable in the forward command on line 6, in place of the integer 63.

Handling other Error Messages

Learning how to program robots isn't easy; it's no different than learning a foreign language. This document covers how to handle some of the more common mistakes and error messages that you may encounter, but you may run into others. That said, here are some general rules for dealing with all error messages:

- Determine if the message is an error, a warning, or just information. The message may not even require additional work on your part!
- Read the error message for clues! Error messages aren't always the most intuitive, but they always contain some information about what the compiler found.
- When your program contains multiple errors, fix them one at a time, recompiling your code after each fix. The "ripple" effect can make it seem like there are errors even if the rest of your program is perfect.
- Pseudocode, pseudocode, pseudocode! Make sure you have a plan in place before you try to write a complex program. That way you can work out the logic first, without having to worry about it and the syntax, spelling and capitalization at the same time.