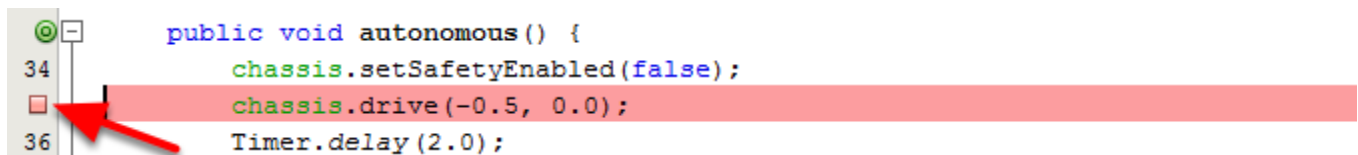


Debugging a Robot Program

Debugging a Robot Program

Debugging the robot program is slightly more complex and can't be used during the competition matches, but can be a very helpful technique for troubleshooting issues with a robot program. Debugging allows you to stop, start and step through the execution of a program and view the values of program variables as you do so. To debug an FRC Java program, first the program has to start, and then you must attach the NetBeans debugger to the running program.

Placing a Breakpoint



Place a breakpoint that you expect to hit by clicking in the gray area to the left of the desired source code line. A breakpoint will cause the code to pause execution when it is reached, allowing the user to view variable values before either resuming execution or stepping through execution one line at a time. You can add additional breakpoints in this step if desired.

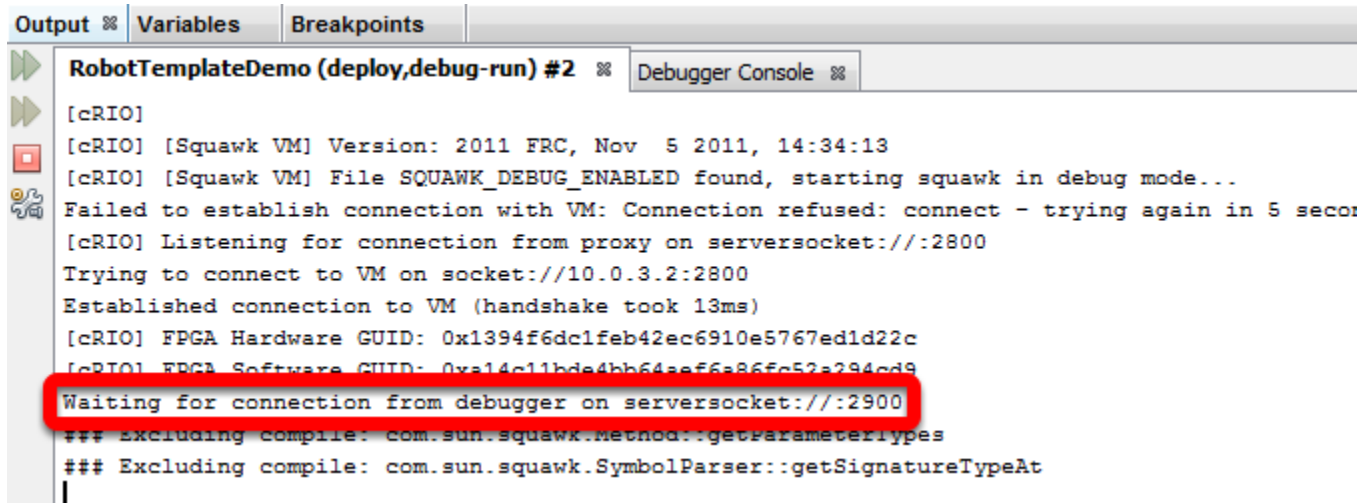
Running the program in Debug Mode



Make sure the program is set as the main program (it will be shown in bold in the left pane, see [here](#) for instructions), then click the Debug button in the toolbar.

Debugging a Robot Program

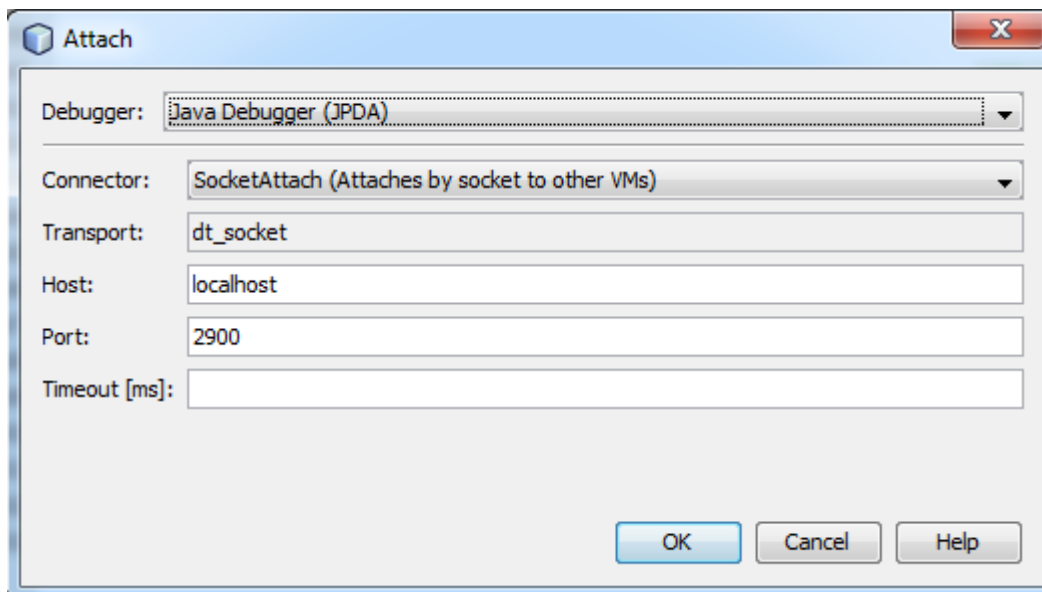
Wait to connect the Debugger



```
RobotTemplateDemo (deploy,debug-run) #2 % Debugger Console %
[cRIO]
[cRIO] [Squawk VM] Version: 2011 FRC, Nov 5 2011, 14:34:13
[cRIO] [Squawk VM] File SQUAWK_DEBUG_ENABLED found, starting squawk in debug mode...
Failed to establish connection with VM: Connection refused: connect - trying again in 5 seco
[cRIO] Listening for connection from proxy on serversocket://:2800
Trying to connect to VM on socket://10.0.3.2:2800
Established connection to VM (handshake took 13ms)
[cRIO] FPGA Hardware GUID: 0x1394f6dc1feb42ec6910e5767ed1d22c
[cRIO] FPGA Software GUID: 0xa14c11bde4bb64aef6a86fc52a294cd9
Waiting for connection from debugger on socket://:2900
### Excluding compile: com.sun.squawk.Method::getParameterTypes
### Excluding compile: com.sun.squawk.SymbolParser::getSignatureTypeAt
```

Wait until the output window displays "Waiting for connection from debugger on socket://:2900". This is when the program will try to connect to the debugger.

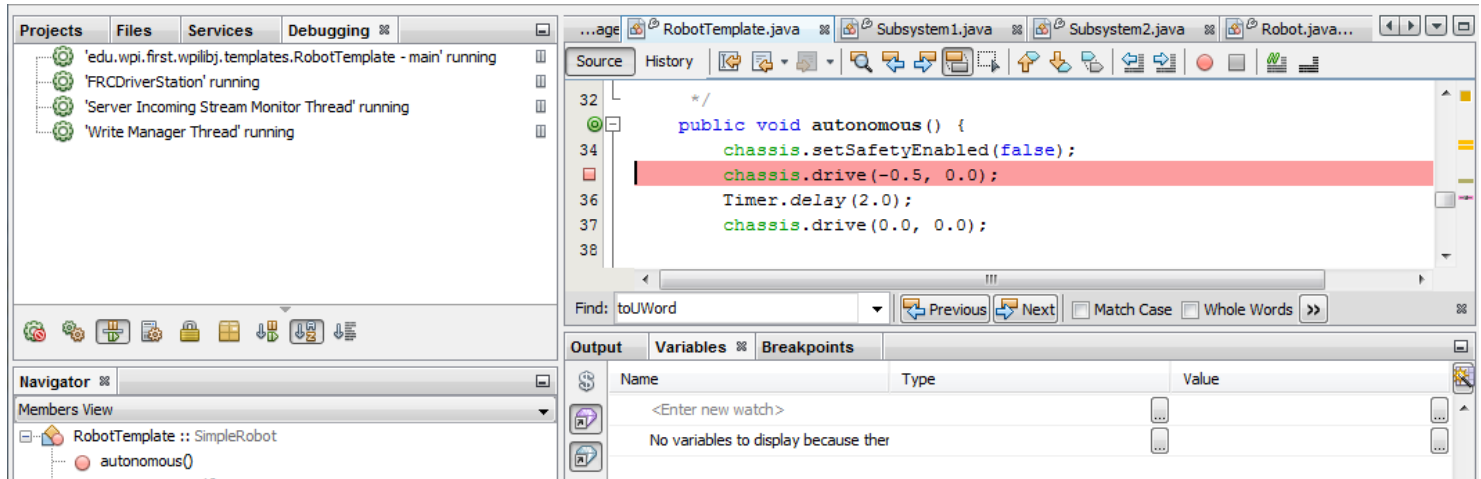
Attach Debugger



Select the Debug menu from the top of the screen and click "Attach Debugger". Make sure the debug options match the ones shown in the picture, then click OK.

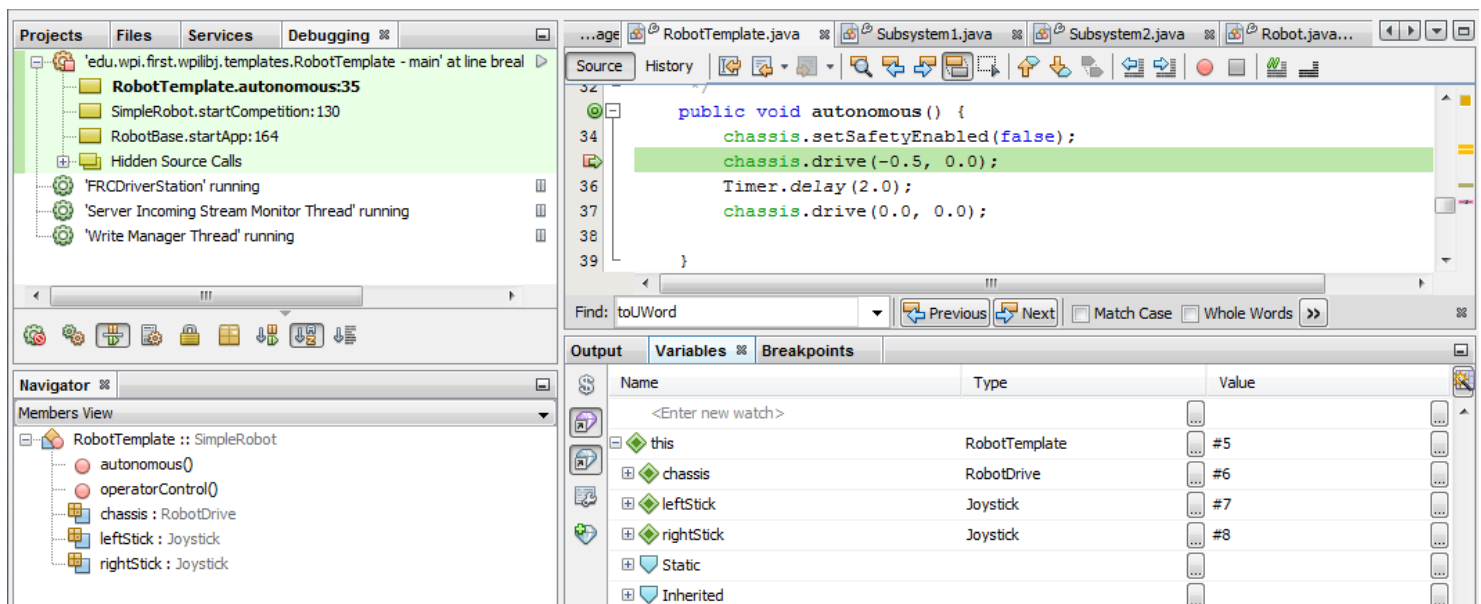
Debugging a Robot Program

Debugger Connected



When the debugger completes the connection, Netbeans should automatically switch the left pane to the Debugging tab and display the running tasks. The bottom pane will switch to the Variables tab which displays variables currently in scope.

Run to Breakpoint

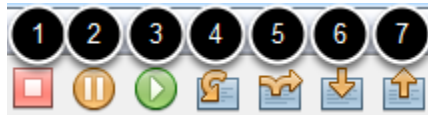


To reach your breakpoint, you may need to connect the Driver Station and enable the robot in the appropriate mode (autonomous, teleop, etc.). When the program reaches a breakpoint, the execution will be paused, the line of code will be highlighted in green, and the Variables tab will be

Debugging a Robot Program

populated with the variables currently in scope. To see all the variables, you may need to expand the tree.

Control Program Execution



After you have reached your breakpoint you can now control the flow of program execution using the buttons in the toolbar or their associated keyboard shortcuts:

1. Finish Debugger Section (Shift+F5) - Terminates the code and closes the debugging connection
2. Pause - Pauses program execution at the current point
3. Continue (F5) - Resumes program execution. The program will execute freely until it reaches a breakpoint or is paused.
4. Step Over (F8) - Steps through one source line, stepping over any method calls.
5. Step Over Expression (Shift+F8) - Steps through one method call in a source line. The value of the method call can then be viewed in the Variables window.
6. Step Into (F7) - Executes one method call in a source line. This will step down into the method.
7. Step Out (Ctrl+F7) - Executes one source-line. If the line is part of a a method, executes the rest of the method and returns to the caller.

You can also set or remove breakpoints while the program is running or stopped at a breakpoint.

Using NetConsole for debugging

Code can also be debugged by using System.out.print statements and receiving them with either the NetBeans console or with NetConsole (**note: do not try to use both simultaneously, only use Netbeans OR the NetConsole at one time**). For more information on using NetConsole see [here](#).