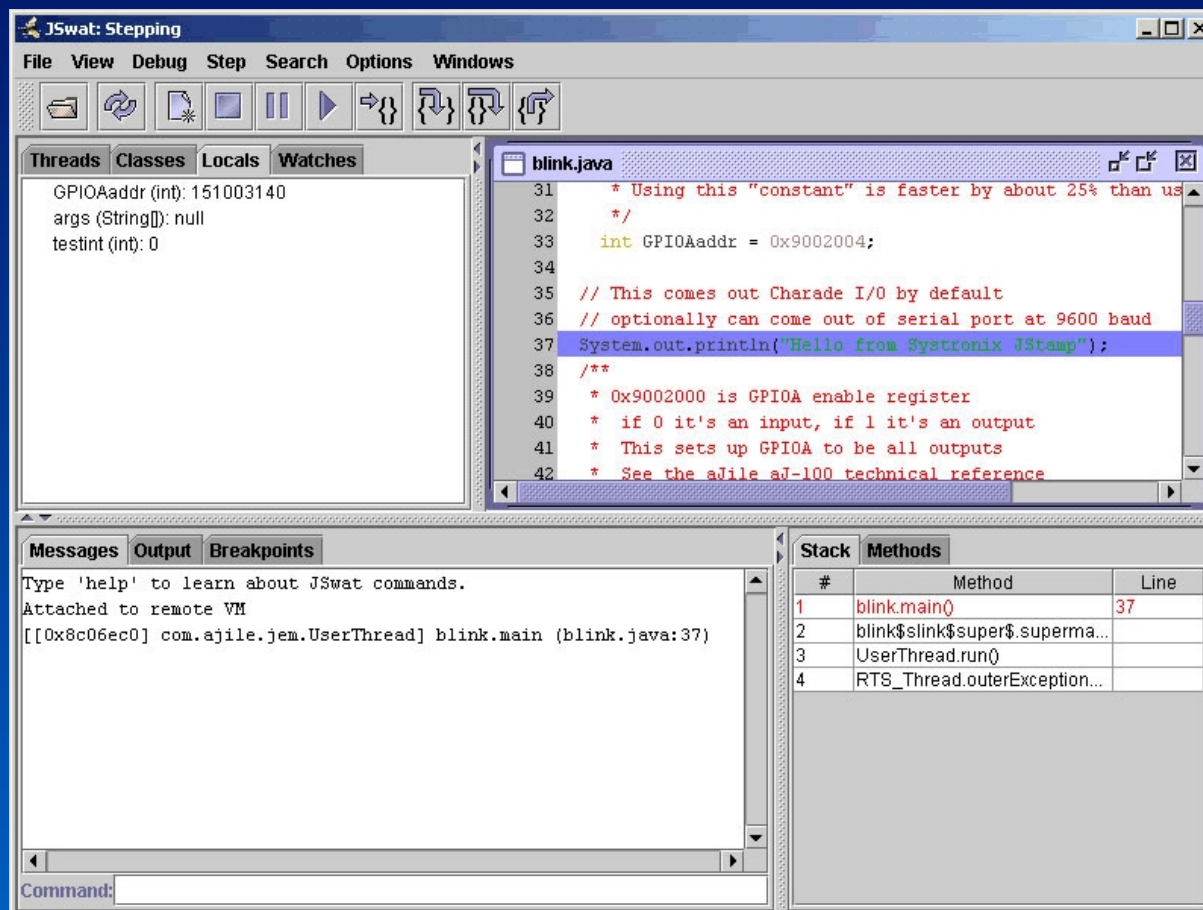


Debugging a JStamp Program

Byte code and Java source level



Charade Help

- Use Charade taskbar help->Commands
- See the `\ajile\charade\Charade Users Guide.pdf`
 - ▶ Acrobat bookmarks are only partially implemented
 - ▶ This has a good macro description
- Try what you think is the command
 - ▶ If you have the name correct, you will get some (terse) help
- Try macros to make your life easier
 - ▶ For example in the `\ajile\charade` folder have some simple “goto” script files such as my “`goto_jstik_ram.sod`”:
 - Execute it by typing `@goto_jstik_ram.sod`
 - Here’s the whole file:
 - `cd J:\PROJECTS\jstik\JStik\mfg\test_code\ProductionTest\JStik\UnitUnderTest\rambuild_jstik_1667014455`



Charade tips

- Don't use whitespace in path names
 - ▶ Charade won't accept these, whitespace is a delimiter
- Charade can only see static variables
 - ▶ You can't watch temps, Charade doesn't know about scope
 - ▶ Charade is case-insensitive
- You can only set breakpoints in RAM code
- When using a flash build you have to load the map manually
 - ▶ Load JVM0.map
- Unless you understand Java byte codes (I don't), breakpoints and stepping will be of limited use.

Charade commands

All entered at the Charade Command line

■ Memory manipulation

- ▶ Set {addr} [=] {data} [data]
- ▶ Display {addr} {addr}
- ▶ Dump {addr} [count]
- ▶ Fill {addr} {addr} {data} (remember data is assumed to be 32 bits)
 - -> fill 0 100 89123456
 - -> du
 - 00000000 = 89123456 89123456 89123456 89123456 V.V.V.V.
 - 00000010 = 89123456 89123456 89123456 89123456 V.V.V.V.

Charade

Polling static variables, setting breakpoints

- Use “show map {field name}” to get the hex address
- Use options->poll->setup to enter the address and start watching
- Breakpoints are similar, but you can set a breakpoint using the method name
 - ▶ For example in the blink program, a Charade command of “Br delay” will set a breakpoint on the blink.delay method:
 - ▶ -> br delay
 - ▶ +[1] break blink.delay(I)V sw -- 0x0002e6f4
 - ▶ vm.0 charadeIO active
 - ▶ vm.0 heap monitor active
 - ▶ vm.0 heap monitor active
 - ▶ **** TARGET HALTED BY S/W BKPT #1 ****

JSwat Debugger

Java Source Level Debugging

- See `\ajile\readme.html`
 - ▶ Click on the “Source Level Debugger” link
- See <http://www.bluemarsh.com/java/jswat/index.html>
 - ▶ This is the JSwat debugger web site
- JSwat debugger demo/walk-through

JSwat Quick Start - part 1

Here are the steps needed to use JSwat with JStamp

- You must be using aJile 3.15 or later
- Include debugging information in your .class file
 - ▶ This is the default in JBuilder, see project-<project properties. Select the “build” tab, “Java” sub-tab, and ensure that Debug Options “source, line, and variable information” is checked.
 - ▶ Or use “javac -g” at a JDK command line
- Tell JemBuilder you are using JSwat
 - ▶ On the taskbar, select Project, then Drivers. JSwat applies to the whole project, not just one JVM, so in the JVM selection box, change this from the default of “JVM0” to “Project”. In the available drivers window you will see a “JemDebug” driver. Click on the right arrow to copy JemDebug into the installed drivers window.
 - ▶ In the JemBuilder (left) navigation pane, expand Project->Drivers. Click on JemDebug and be sure that the port (0x378 for PC LPT1) and paths are correct.
 - ▶ Save the project definition with these changes and rebuild.
 - ▶ You will see added lines in the output window, building jemDebug.bat and jswat.bat

JSwat Quick Start - part 2

- Close Charade if it is open. JSwat replaces it.
- Start the JTAG JemDebug proxy
 - ▶ Browse to your JemBuilder project output folder, in my case “rambuild”
 - ▶ Double click on jemDebug.bat - this will open a DOS command window and load your project binary file onto JStamp via the JTAG port. Lots of progress data will display. The last line should be “waiting for connection.” This is the proxy waiting for connection to JSwat.
- Start JSwat
 - ▶ Double click (in the same project output folder) on jswat.bat which will open another DOS window and then start the jswat GUI.
 - ▶ Click the JSwat task bar Debug->Attach to remote. Use “localhost” and “12345” for the Host and Port settings. Click OK.
 - ▶ There should now be a list of threads on the thread tab. Select thread 1, which should be your user thread. Now the JSwat Methods, Stack, and Locals tabs should be populated with values from your source code.

JSwat Quick Start - part 3

Start stepping

- Try stepping into your program and viewing your source code
 - ▶ use the Taskbar File->Load or
 - ▶ Click on one of the step buttons (the arrow/curly brace icons)
 - ▶ If you get the error message “Could not find source file for class. (Method: ...” JSwat can’t find your .java files. Tell it where to go in the taskbar Options->Set Sourcepath dialog. This is the path to your .java files, not your .class files