Embedded Java IDE

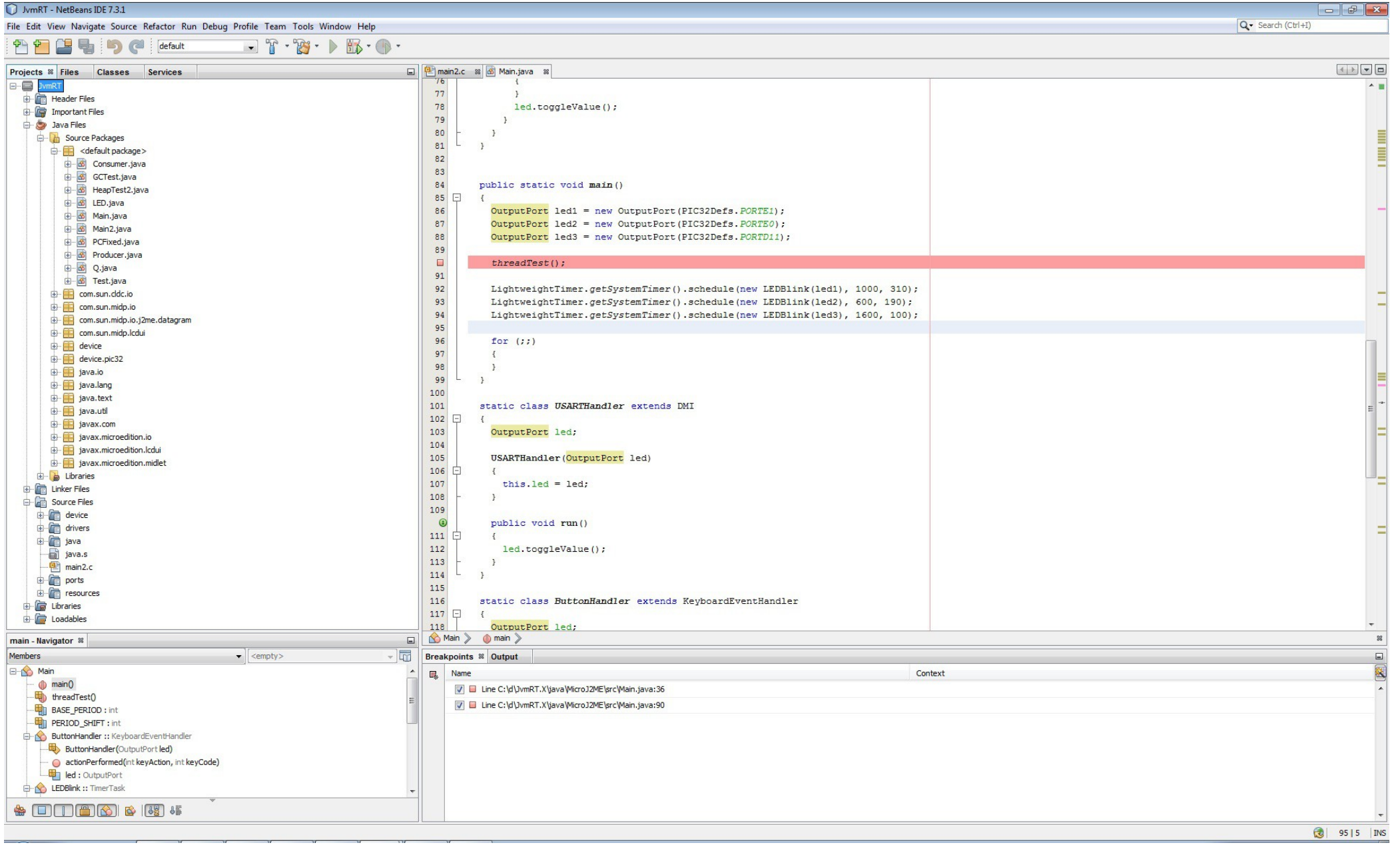
Embedded Java IDE is a plugin for NetBeans 7 framework. With some moderate efforts it can be converted into MPLAB X IDE extension.

Embedded Java project

Embedded Java project is an extension of conventional MPLAB X project, compatible upside down. We can see additional folder “Java Files”, which contains Java subproject with Java sources.

We can open MPLAB X project in this IDE , of course without Java part. And we can open Embedded Java projects in MPLAB X IDE, in this case Java subfolder will be represented as ordinary file tree and during debugging all Java structures will be represented as their C counterparts.

Embedded Java project

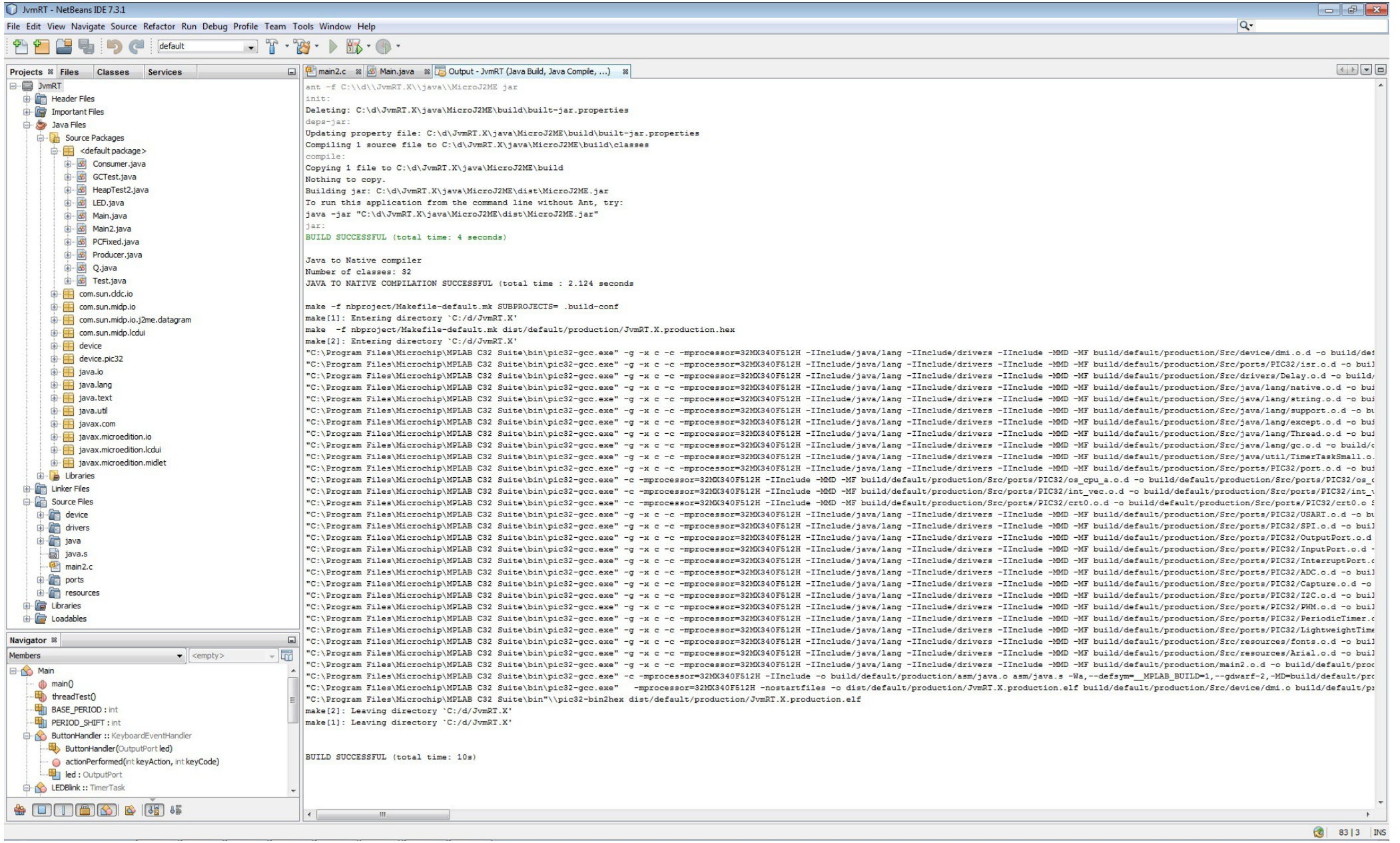


Project Build

Then we can build entire project. In this figure we can see all stages of building process:

- Build Java subproject
- Java to Native (MIPS assembler) compilation
- Build Native (C & asm language subproject)

Project Build



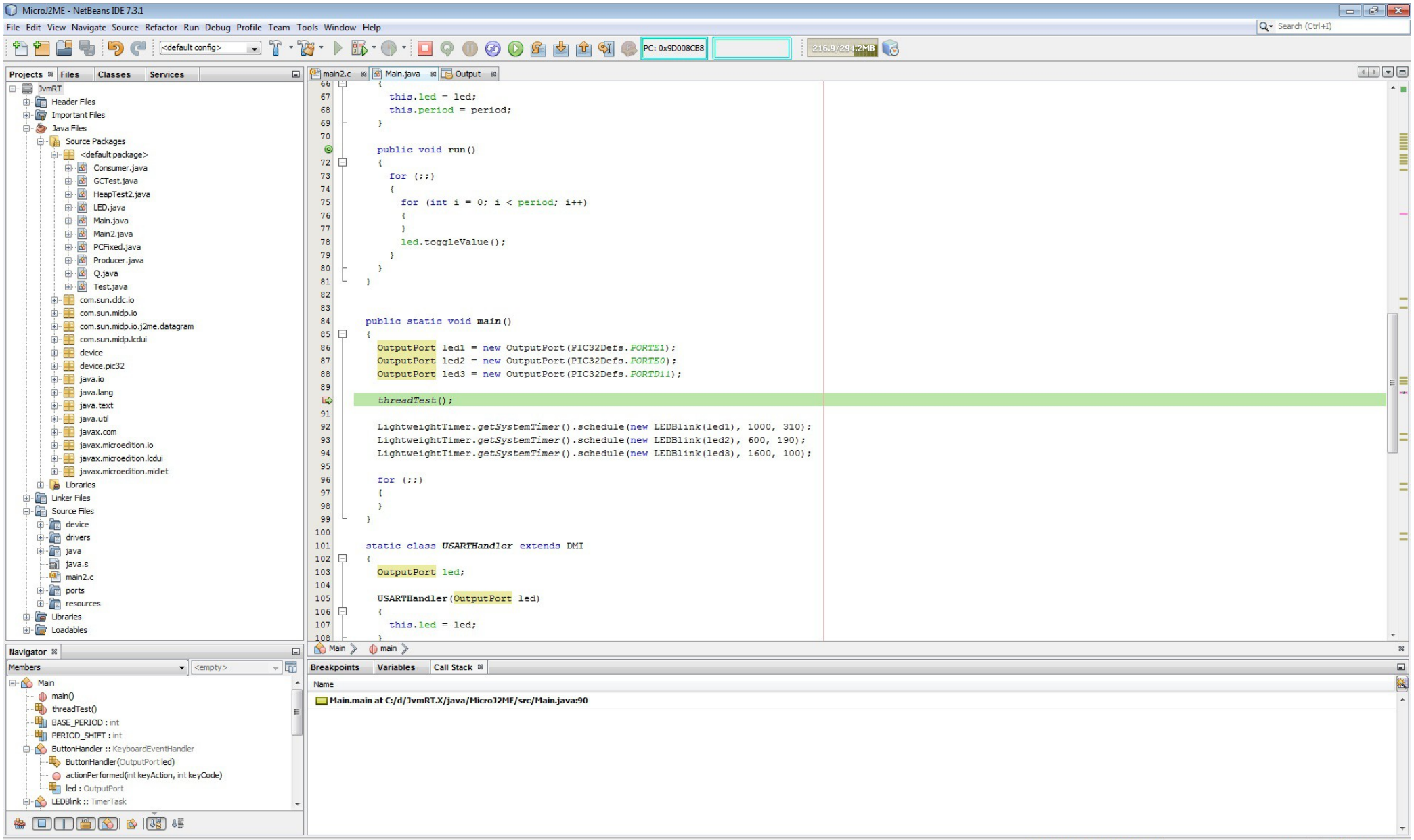
Board connection

After build, real board with PIC32 can be connected. *Current version supports PicKit 2 and PicKit 3 adapters, this list can be easily extended*.

PIC32 will be programmed, and debugger starts.

Program stops on breakpoint from Java source file.

Board connection

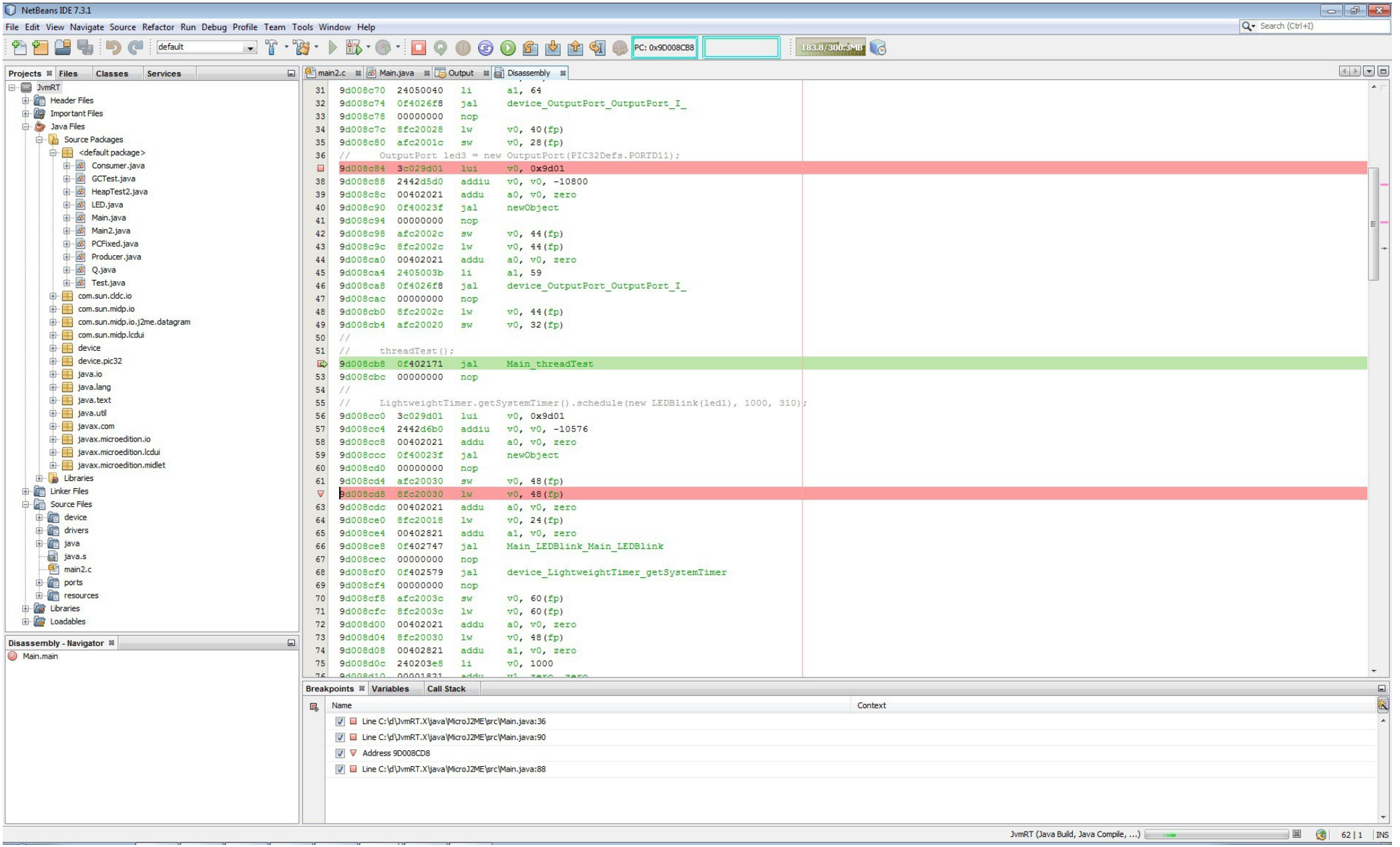


Disassembler Window

Now we can switch to Disassembler Window and see PIC32 assembler code *(this is code with maximum debugging capabilities, not for evaluation of compiler optimizations).*

In Disassembler Window we can toggle breakpoints of two different types – Java/C Source line breakpoints and Address breakpoints, system automatically choose which type of breakpoint will be toggled. If disassembly line has mapping from Java/C source, then Source line breakpoint will be added/removed, otherwise – Address breakpoint, which can be seen only in Disassembler of Breakpoints window

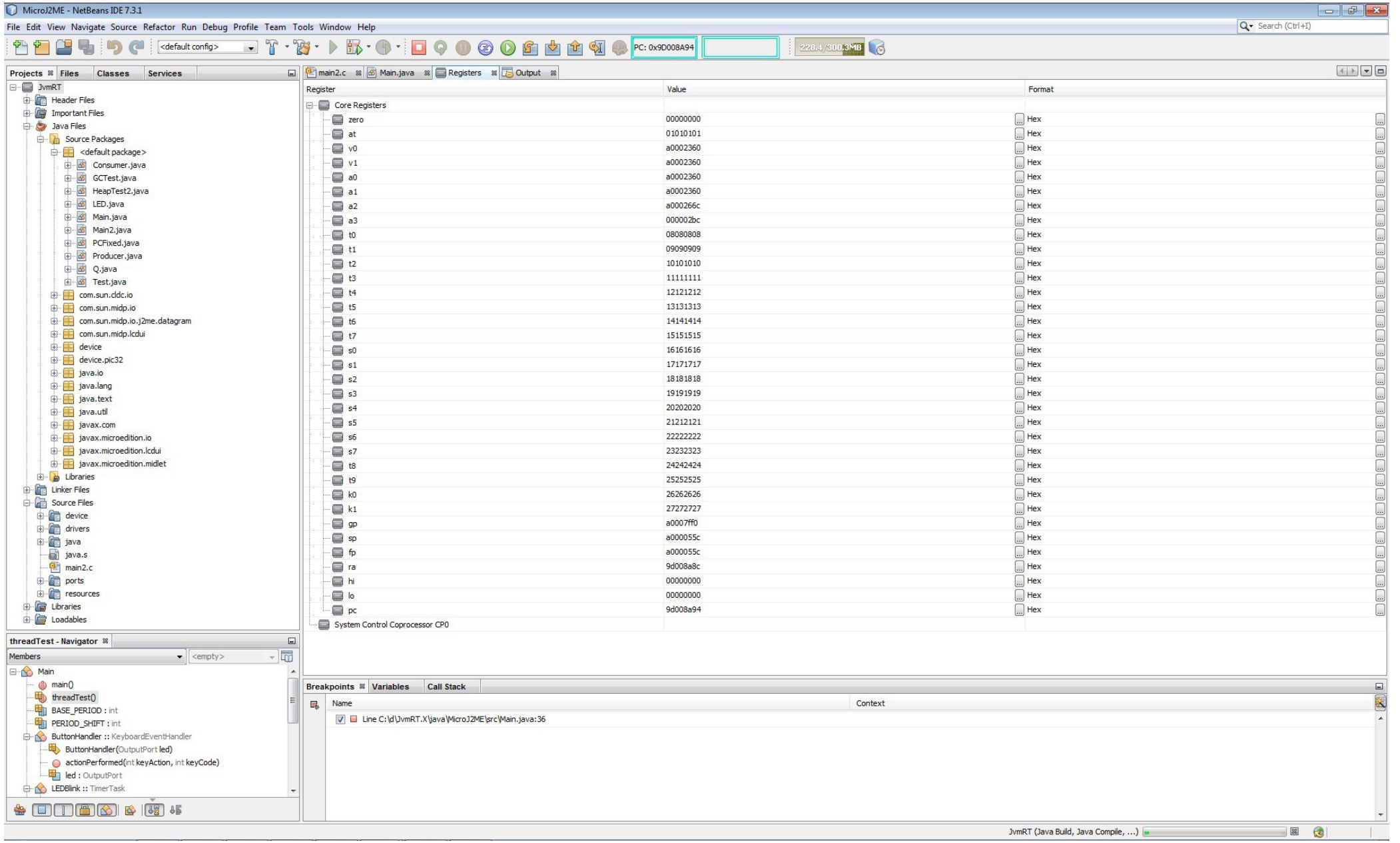
Disassembler Window



Processor registers window

Processor registers window contains nothing special, its slightly more convenient than in MPLAB X because it can tree representation of different register groups (peripherals included).

Processor registers window



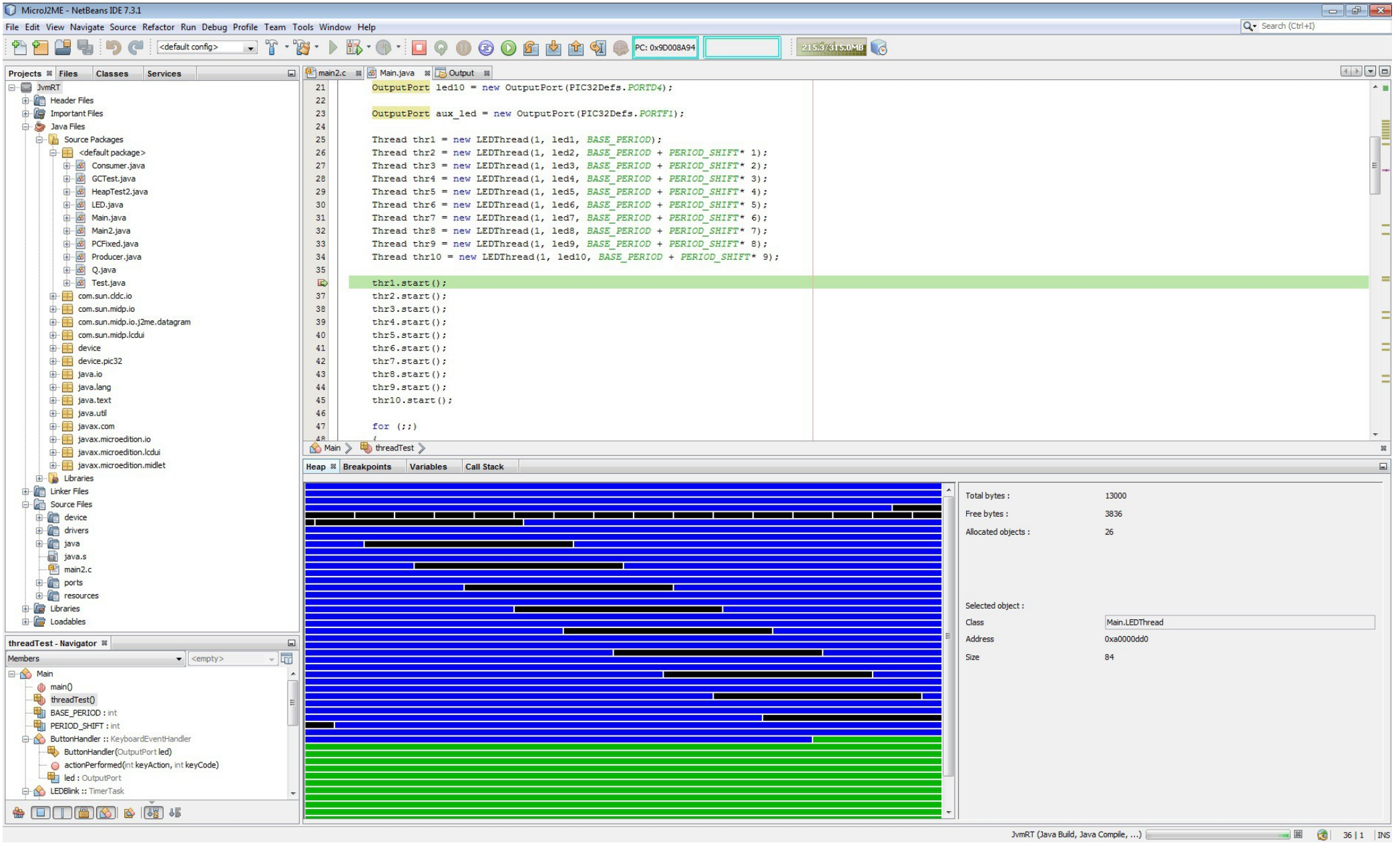
Runtime Heap Window

Some useful window, which represent runtime Heap.

- Green color – free memory blocks.
- Black color – dynamically allocated Java objects
- Blue color – special allocated regions (for example, processor thread stacks).

If we click on some black region, we can see some information of allocated object (address, size and Java Class of this object)

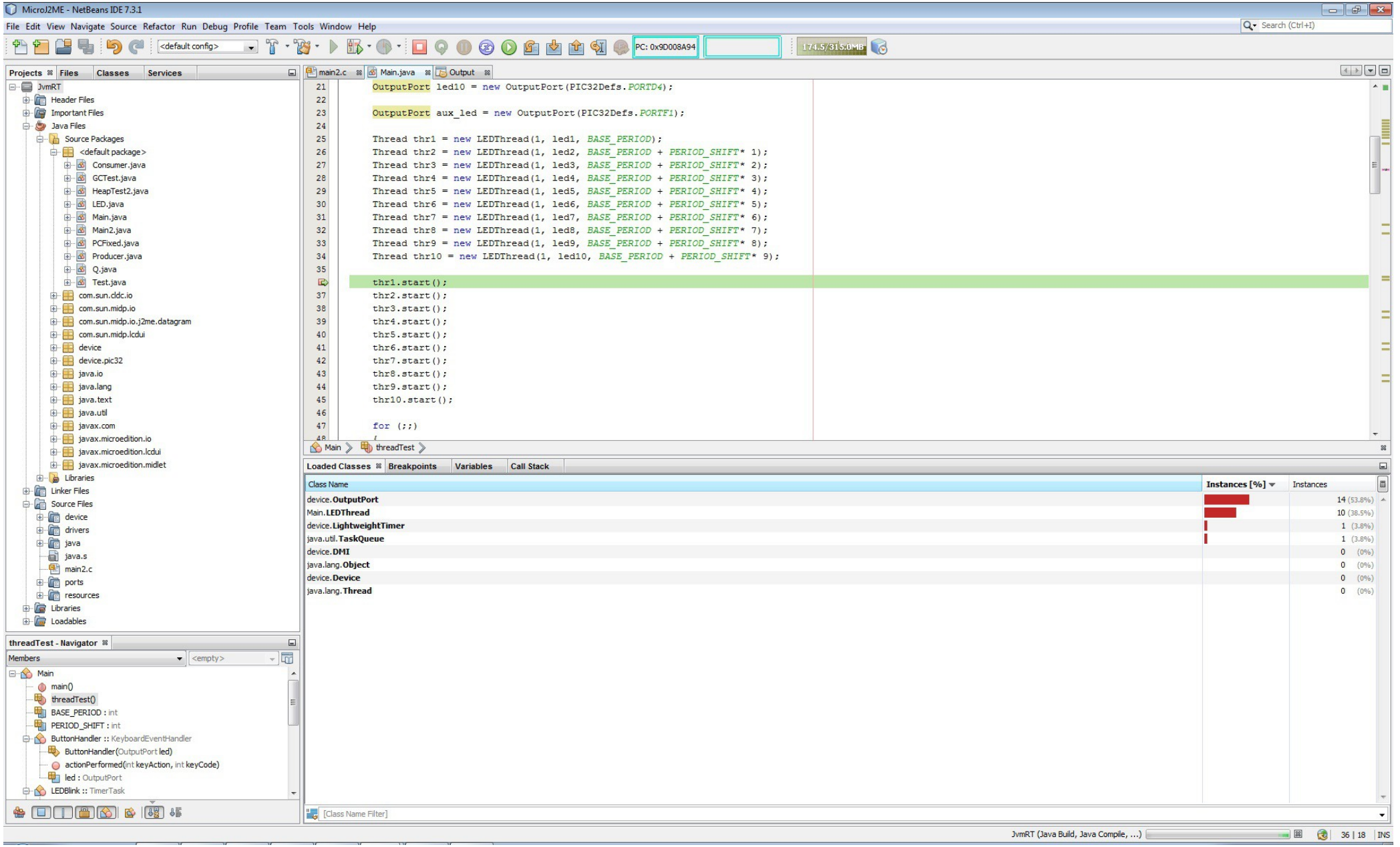
Runtime Heap Window



Runtime Status

Another useful information about runtime status. We can see how many object of which Java class are allocated in this execution point.

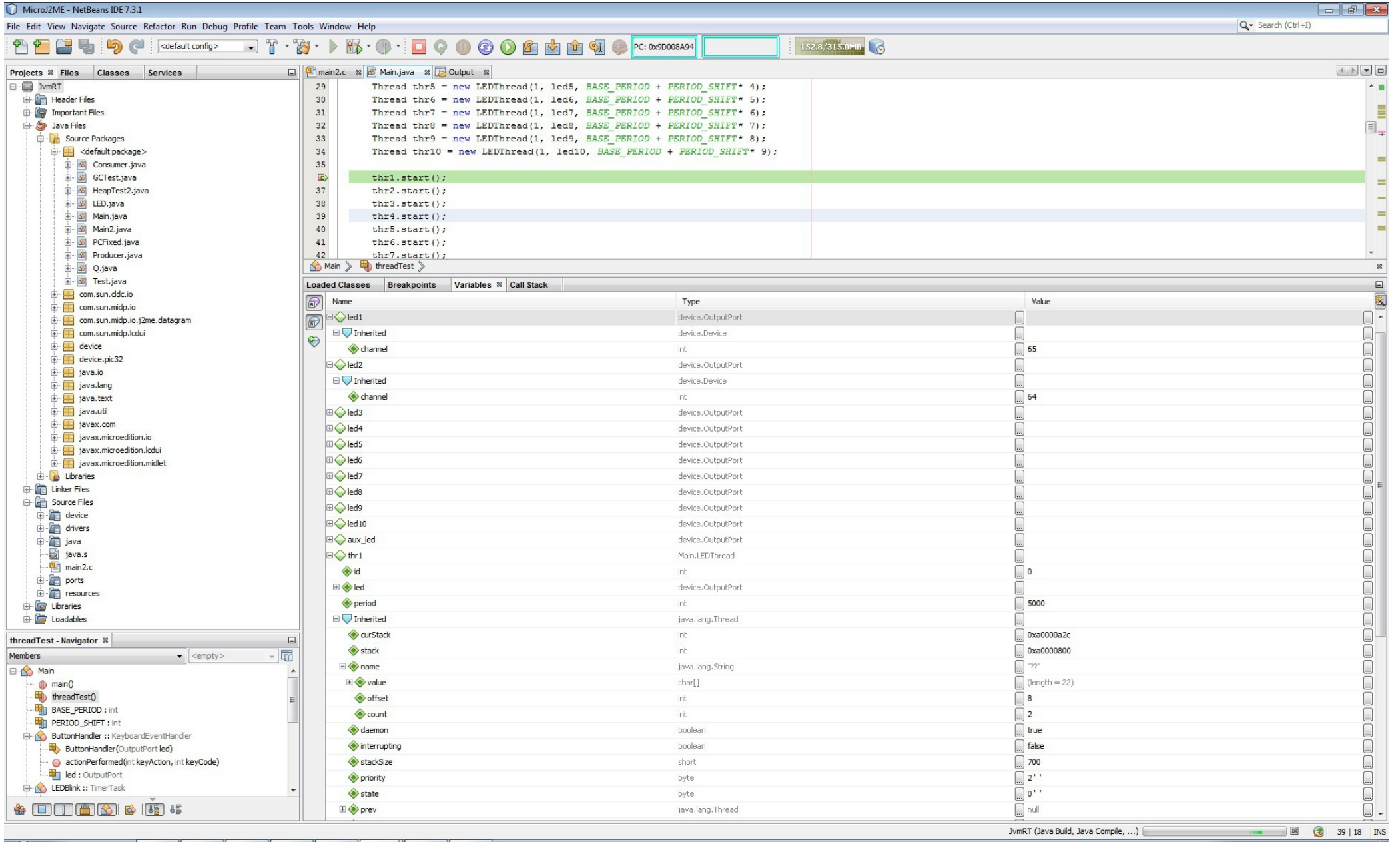
Runtime Status



Variables and Watches Windows

Variables and Watches Windows contain nothing special. System will show variable according with current execution point. If we stopped on Java source, all variables and watches will be Java types and classes. If we stopped on C source, there will be native C structures or native counterparts to Java classes.

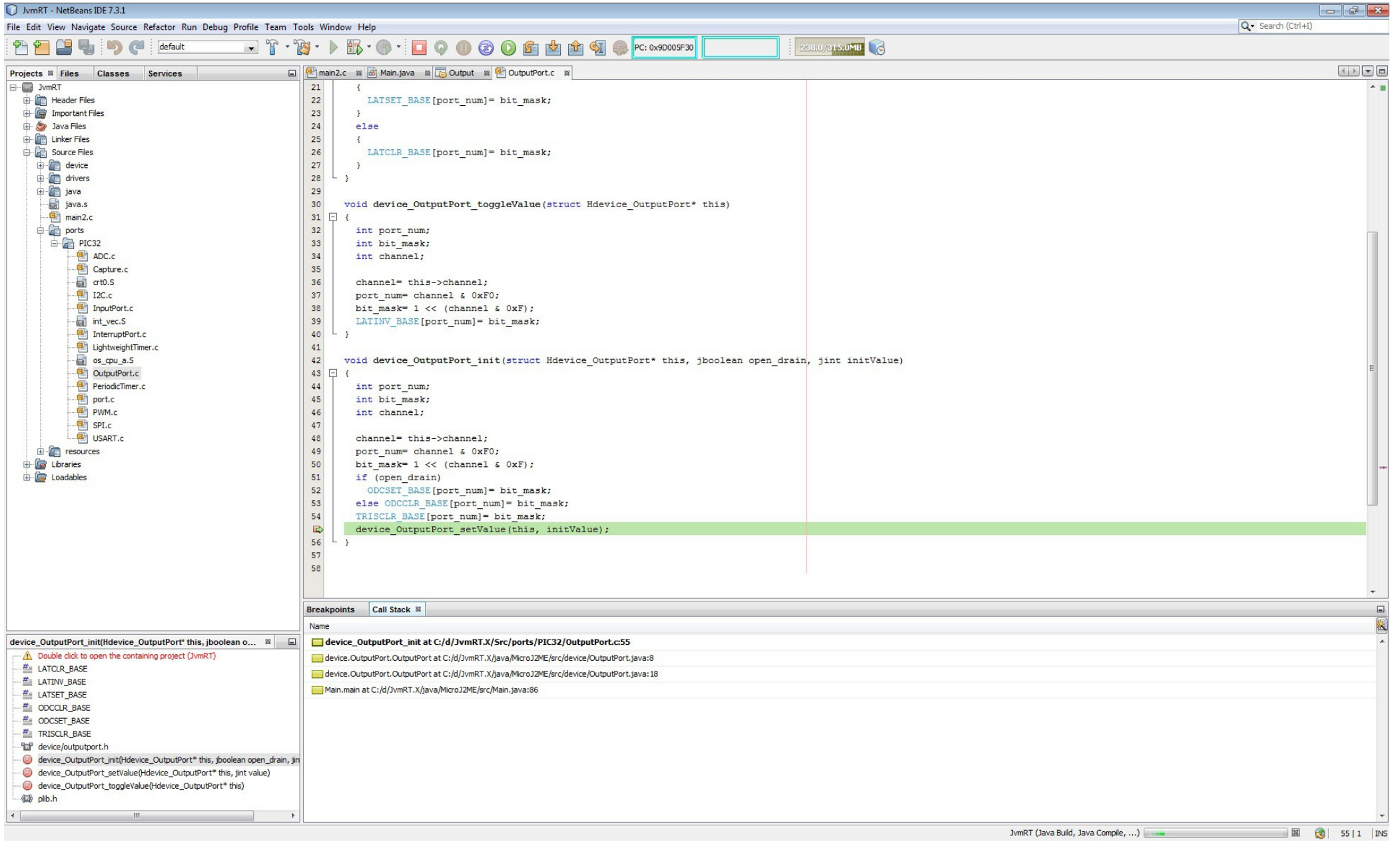
Variables and Watches Windows



Native Methods Debugging

We stopped on Java native method implementation. Call Stack Window reflects this situation: it shows Java methods and also native methods and C functions.

Native Methods Debugging



Native Methods Debugging

Then we can open original Java class source from which its native method was implemented and see some methods in current calling sequence.

Native Methods Debugging