Java Environment for a JUMPtec WebToNet Embedded System

Valter Filipe Silva, José Alberto Fonseca, José Luís Oliveira
Polytechnic Institute of Castelo Branco, University of Aveiro

Workshop on Java for Embedded Systems, 2002

University of Karlsrhue, Germany

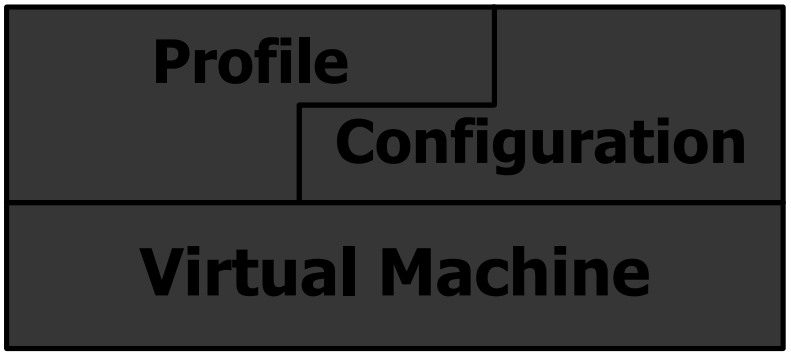
Objectives

- To execute programs written in Java in a small JUMPtec embedded system.
- To help the development of applications for the control aplications.
- To evaluate the Java language performance in this kind of devices.
- To evaluate the possibility of using mobile code in these systems.

Contribution of Java

- Ideal situation, Write Once Run Anywhere...
- Must port the virtual machine for all these platforms.

Java 2 Micro Edition architecture



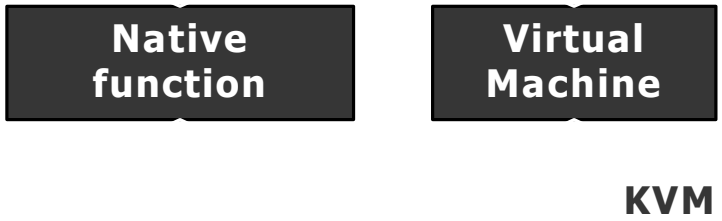
- Java 2 Micro Edition:
  - Two virtual machines
  - Configurations (CDC, CLDC)
  - Profiles (ex. MIDP)
- Configuration
  - Defines the basic environment:
    - Virtual machine
    - Minimal classes
- Profile
  - Oriented to a specific device

➢ **Great flexibility and possibility of evolution**

KVM (*Kilo Virtual Machine*)

- Available with CLDC configuration.
- New implementation of the standard java virtual machine
- Executes the same set and format of Byte Codes than a standard virtual machine.
- Small and modular to satisfy a great number of constrained resource devices.

KVM – Native Functions

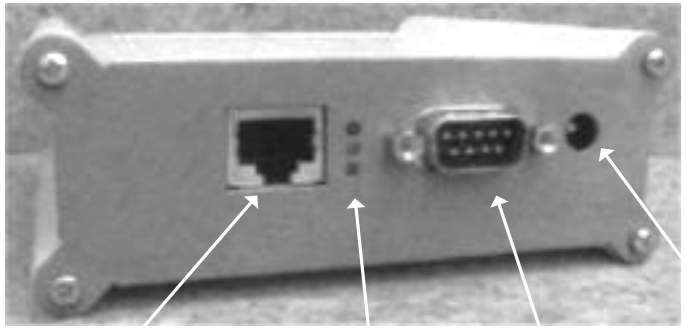


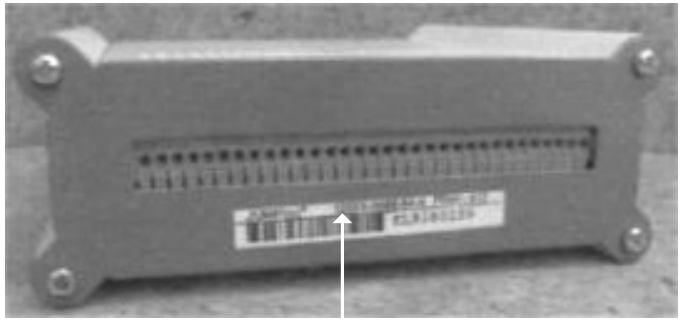
- Different from JNI in J2SE
- Protect the system, but give access to it.
- Written in C.
- Integrated in the Virtual Machine.

Preverification

- Verification divided in two phases in two different systems.
- It is necessary to pre-verify the code outside the target device.
- Better performance and less memory usage

The JUMPtec WebToNet





- Small
  - 11.2*11.6 cm
- 386SX Processor
- 2 MB of RAM
- 2 MB of flash memory
- Integrates a Web server.
- Operating system
  - DR-Dos

The KVM in DOS

- Configuration of source code flags.
- Functions to support 64 bits arithmetic.
- Alterations to the environment to support floating point.
- Native functions to access to the hardware.
- Modification of the Class Loader.
- Compilation of the source code with a 32 bits pointers compiler.

Class Loader problem

Class name	DOS name
My_program.class	My_pro~1.cla
My_program1.class	My_pro~2.cla

- Solution: The Class loader must verify the class name in its internal structure

- The file names in DOS have 8+3 characters.
- In Java, the class name can have more than 8 characters.
- The name of file doesn't reflect the class name.

Floating Point arithmetic

- KVM originally doesn’t provide this support.
- Many applications of this kind of systems use FP arithmetic (e.g. controllers).
- Changes to the environment:
  - KVM, necessary to add some functions
  - KVM, necessary to add native functions
  - Also necessary:
    - To add some classes (namely `float` and `double`)
    - To add some methods to existing classes (e.g. `println`)

Native Functions

- Native functions necessary to provide floating point facilities.
- Native Functions to access the hardware ports:
  - Similar to `inportb()`.
  - Similar to `outportb()`.
- Native functions to access to the ethernet card.

Interaction with the real world

- Serial Port
  - Developed a class with several static methods for common tasks.
- Parallel port
  - Developed a class with some important methods for standard access to the parallel port.
- Programmable LED
  - Static class with two methods, to turn it on and turn it off for debugging purposes.
- Network access
  - Operating System: Dos → Packet Driver must be used.
  - Ethernet access is made with *wattcp* (www.wattcp.com), with the Generic Connection Framework and some native methods.

Application Example

- Send via rs232 an IP address
- The JUMPtec system checks in the DNS the corresponding name.
- The JUMPtec sends via serial port the corresponding host name.
- After, the LEDs are turned on and off according to the switches' state.

Java Performance

- The application to turn on and off the programmable LED.

Programming language	Execution time	Tool
Assembly	14 μS	Masm 6.11
C	14.8 μS	Turbo C 2.0
C++	47 μS	Gnu C 2.95
Java	640 μS	javac

- The Java is about 10 times slower than C++:
  - This performance is acceptable for most of the applications of this kind of systems.
- The source code for java is more structured.

PID controller execution time

language	Arithmetic	JUMPtec	PC
Java	Floating Point	14 mS	33.5 μS
	Integer	1.28 mS	25 μS
C	Floating Point	12.5 mS	8.6 μS
	Integer	65 μS	0.6 μS

- Floating point unit is important...
- Java is slower, but with floating point, has similar performance

Real Test

- Control of a kiln temperature with the JUMPtec system.
- Control algorithms tested:
  - PI
  - On Off
  - Ramp
- Easy to develop programs, namely to control the sampling period.

Possible Utilizations

- Programming paradigm changed for embedded system.
- Simplified Development.
- Applications
  - Instrumentation.
  - Control applications.
  - Systems for teaching purposes.

Mobile Agents

- Serialization
  - KVM doesn’t support serialization.
- User defined Class Loaders
  - Just one class loader is allowed in KVM, the original.
- More tests with the system using network.

Conclusions

- Rapid develop of applications.
- Code portability and platform independence.
- Non proprietary system.
- Facilities for use in control systems.
- In the future we hope use mobile agents in this platform.

Thanks for your attention

Questions

vfs@est.ipcb.pt