

Trampoline:
an open platform
for (small) embedded systems
based on OSEK/VDX and AUTOSAR

<http://trampoline.rts-software.org/>

Jean-Luc Béchenec^{1,2}, **Sébastien Faucou**^{1,3}

¹IRCCyN (Institute of Research in Communications and Cybernetics of Nantes)

²CNRS (National Center for Scientific Research) / ³University of Nantes

10th Libre Software Meeting

What is Trampoline?

Trampoline is an open platform for small embedded systems with real-time constraints

web site: `http://trampoline.rts-software.org`

svn: `http://trampoline.rts-software.org/svn/trunk/`

forum: `http://trampoline.rts-software.org/bb/`

It is inspired by the OSEK/VDX OS and AUTOSAR OS standards

It is composed of

- ▶ a real-time kernel (trampoline): LGPL (+ BSD)
- ▶ an off-line kernel configuration tool (goil): GPL
- ▶ a virtual environment to prototype applications on workstations (viper): LGPL

OSEK/VDX? AUTOSAR?

OSEK/VDX: Öffene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug / Vehicle Distributed Executive

AUTOSAR : Automotive Open System Architecture

OSEK and AUTOSAR are group of companies of the automotive industry that want to establish standards for in-vehicle embedded software components:

- ▶ RTOS,
- ▶ communication protocols,
- ▶ device drivers,
- ▶ etc.

Main characteristics of in-vehicle embedded systems

Closed, static systems:

- ▶ all objects (threads, mutexes, queues, etc.) can be created at compile-time

Bare hardware (to keep production costs as low as possible):

- ▶ 16 or 32 bit CPU, freq down to 20 MHz
- ▶ RAM down to 32 KB
- ▶ ROM (Flash) down to 128 KB

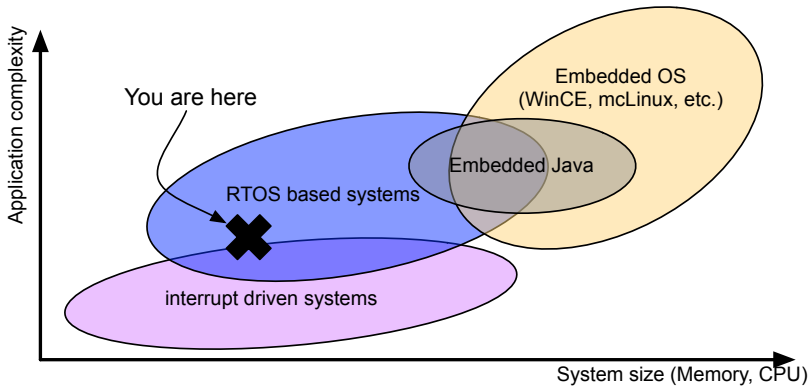
Application=mixed digital (time driven) and reactive (event driven) control:

- ▶ need for real-time predictability

Cyber-physical systems:

- ▶ support for interrupt-driven computations

Trampoline has been designed to support all these requirements



Trampoline ...

is well-suited for small "cyber-physical" systems:

- ▶ digital controllers for engine, brakes, wheels, ... of road vehicles
- ▶ small robots (port to Lego[®] Mindstorms[®] in progress)
- ▶ small drones (UAV, "eels", etc.)
- ▶ motes (aka nodes in a wireless sensor network)
- ▶ etc.

is not adapted to general mobile computing systems:

- ▶ No creation of object at run-time
- ▶ No filesystem
- ▶ No TCP/IP connectivity (not yet)

Why is it named Trampoline?



Why is it named Trampoline?

In embedded systems, "trampolines are short snippets of code that start up other snippets of code". Rather than write interrupt handlers entirely in assembly language, another option is to write interrupt handlers mostly in C, and use a short trampoline to convert the assembly-language interrupt calling convention into the C calling convention.¹

¹ *Trampoline (Computers) - Wikipedia, the free encyclopedia*, [http://en.wikipedia.org/wiki/Trampoline_\(computers\)](http://en.wikipedia.org/wiki/Trampoline_(computers)), retrieved July 3rd, 2009.

Why is it named Trampoline?

In embedded systems, "trampolines are short snippets of code that start up other snippets of code". Rather than write interrupt handlers entirely in assembly language, another option is to write interrupt handlers mostly in C, and use a short trampoline to convert the assembly language interrupt calling convention into the C calling convention.¹

Well, the truth is: the project has known many ups and downs!

¹ *Trampoline (Computers) - Wikipedia, the free encyclopedia*, [http://en.wikipedia.org/wiki/Trampoline_\(computers\)](http://en.wikipedia.org/wiki/Trampoline_(computers)), retrieved July 3rd, 2009.

Present state of the project

- ▶ A core team of 5 developers working actively on the project, with some external contributors
 - ▶ around 40 commits in june 2009
- ▶ A small community communicating through a forum (<http://trampoline.rts-software.org/bb/>)
- ▶ Version 1.1 released in february 2008 successfully passes OSEK conformance test suite
- ▶ Version 2.0 shall be released in october 2009
- ▶ Different versions are presently in use (teaching, R&D, but nothing in production)
- ▶ Available ports: POSIX, Freescale PPC and S12(X), ARM (7, 9, 11), Infineon C166, Atmel AVR8, Nec V850e

Who is using Trampoline?

EE/CS department of universities in Nantes, Rennes, Angers, Toulouse

- ▶ Teaching (labs in embedded and real-time systems)
- ▶ Research
- ▶ In both cases, an open source project is a good choice (freedom to study and modify)

Car makers (Renault, PSA), suppliers (Geensys), software editors (Trialog, Geensys)

- ▶ Internal R&D projects, internal training
- ▶ Some are interested by the freedom of use as in free beer
- ▶ Some others are interested by the freedom of use as in free speech
- ▶ None plan to use Trampoline in production systems for the moment: how to use OSS in critical systems?

+ Some anonymous users from Germany, India, etc. (from the web site logs)

Main characteristics of Trampoline

- ▶ support for real-time multithreading
- ▶ small memory footprint
- ▶ support for early development stages on POSIX workstation within a virtual environment
 - ▶ my IDE for early stages: bash+vim+make+gdb+svn :)
- ▶ compatible with OSEK/VDX OS and AUTOSAR OS API
- ▶ portable (clear separation between hardware dependent and hardware independent code)
- ▶ work-in-progress:
 - ▶ kernel support for multicritical systems (memory and timing protection)
 - ▶ scriptable (w/ python) virtual execution environment (viper2)
 - ▶ logging and post-mortem analysis (T³: Trampoline Trace Toolkit)

Main characteristics of Trampoline

- ▶ **support for real-time multithreading**
- ▶ **small memory footprint**
- ▶ **support for early development stages on POSIX workstation within a virtual environment**
- ▶ compatible with OSEK/VDX OS and AUTOSAR OS API
- ▶ portable (clear separation between hardware dependent and hardware independent code)
- ▶ work-in-progress:
 - ▶ kernel support for multicritical systems (memory and timing protection)
 - ▶ scriptable (w/ python) virtual execution environment (viper2)
 - ▶ logging and post-mortem analysis (T³: Trampoline Trace Toolkit)

Support for real-time multithreading

3 types of threads are supported

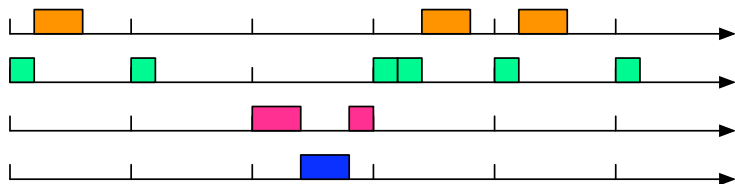
- ▶ ISR: hardware triggered threads with run-to-completion semantics
- ▶ Basic tasks: software triggered threads with run-to-completion semantics
- ▶ Extended tasks: software triggered threads with blocking synchronization points (kind of finite state machine)

Deterministic preemptive scheduling with fixed priorities and preemption threshold

- ▶ once activated, a thread competes for the CPU with its priority
- ▶ once started, it competes with its preemption threshold (\geq priority)
- ▶ when a thread executes a critical section, its preemption threshold is raised

Support for real-time multithreading (cont'd)

Thread	Arrival	Exec time	Priority	Threshold
orange	@0 then every 10	2	1	4
green	@0 then every 5	1	2	4
pink	@10	3	3	4
blue	@12	2	5	5



It is possible to compute the worst-case response time of every thread (ref. on demand)

Support for real-time multithreading (cont'd)

The threshold mechanism is also used to ensure "real-time friendly" mutually exclusive access to shared resources within critical section

- ▶ Known as "Highest Locker Protocol"
- ▶ Preemption thresholds of each resources is computed off-line by Goil
- ▶ Avoids deadlocks (remember starving philosophers?) and priority inversions (aka "Mars Pathfinder" bug)

Support for real-time multithreading (end)

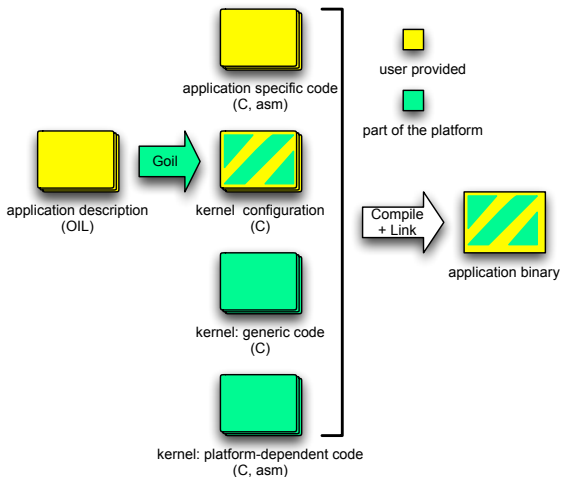
Trampoline API offers a limited number of abstractions, to "ease" the programming of real-time multithreaded program

But don't get fooled: it's still a tricky task

Main characteristics of Trampoline

- ▶ ~~support for real-time multithreading~~
- ▶ **small memory footprint**
- ▶ **support for early development stages on POSIX workstation within a virtual environment**
- ▶ compatible with OSEK/VDX OS and AUTOSAR OS API
- ▶ portable (clear separation between hardware dependent and hardware independent code)
- ▶ work-in-progress:
 - ▶ kernel support for multicritical systems (memory and timing protection)
 - ▶ scriptable (w/ python) virtual execution environment (viper2)
 - ▶ logging and post-mortem analysis (T³: Trampoline Trace Toolkit)

Small memory footprint



The build process:

1. goil generates the kernel configuration (and the makefile)
2. the compiler + linker generates the binary file

Small memory footprint (cont'd)

All descriptors are stored in static variables, initialized at compile time

- ▶ All static fields can be mapped in ROM
- ▶ All dynamic fields have to be mapped in RAM

	basic task / ISR	extended task	alarm	resource
static de- descriptor	4 ptr + 4 uint8		3 ptr + 2 uint8	–
dynamic descriptor	1 ptr + 4 uint8	1 ptr + 6 uint8	3 ptr + 4 uint8	2 ptr + 2 uint8

- ▶ Could be improved by using bitfields instead of uint8, but portability would be decreased
- ▶ Simple applications run on 8bit CPU, RAM < 2KB, ROM < 8KB

Small memory footprint (cont'd)

Simple RAM optimization: take advantage of the preemption threshold mechanism

- ▶ when 2 threads have the same preemption threshold, their execution cannot overlap
- ▶ thus, they can share the same context (including the stack)

More aggressive RAM optimization: context sharing between threads whose execution shall not overlap based on timing information (activation law and execution time)

- ▶ requires a good knowledge of the system
- ▶ should not be used without a timing protection mechanism

The simple optimization shall be included in Trampoline 2.0. The development of a timing protection mechanism is in progress.

Main characteristics of Trampoline

- ▶ ~~support for real-time multithreading~~
- ▶ ~~small memory footprint~~
- ▶ **support for early development stages on POSIX workstation within a virtual environment**
- ▶ compatible with OSEK/VDX OS and AUTOSAR OS API
- ▶ portable (clear separation between hardware dependent and hardware independent code)
- ▶ work-in-progress:
 - ▶ kernel support for multicritical systems (memory and timing protection)
 - ▶ scriptable (w/ python) virtual execution environment (viper2)
 - ▶ logging and post-mortem analysis (T³: Trampoline Trace Toolkit)

The POSIX port

Fact 1: using the development chain of a given embedded target can sometimes be painful

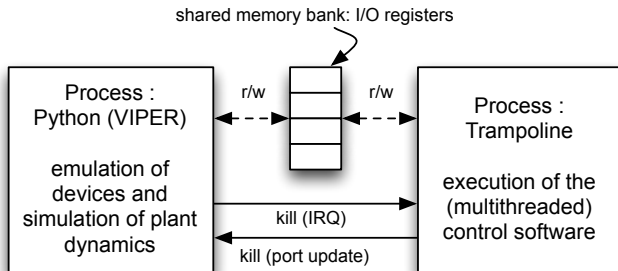
Fact 2: we are used to develop on workstation (Eclipse, Vim+make, etc.)

1 + 2 = 3: we want to be able to perform the early prototypes on a workstation, with our usual tools

That's the purpose of ViPER (Virtual Processor Emulator)

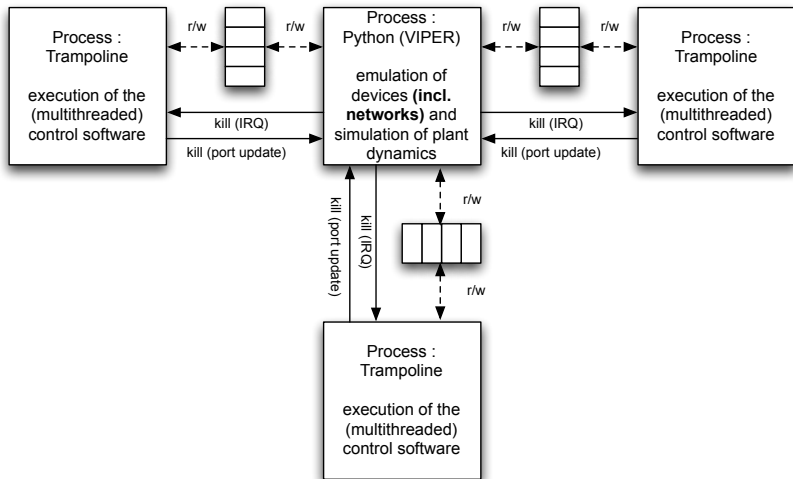
- ▶ The following slides concern the new POSIX port and ViPER2 (both will be included in Trampoline 2.0)

The POSIX port (cont'd)



- ▶ VIPER emulates the behaviour of the devices (including timers) and simulates the dynamics of the controlled plant
- ▶ Trampoline runs the multithreaded control software
 - ▶ same code, except for platform-dependent parts: boot, context switch (100% POSIX C code), low-level device access
- ▶ POSIX signals and shm are used to emulate the communication between the CPU and the devices

The POSIX port (cont'd)



The virtual environment will emulate networks ! (work in progress, shall be integrated in Trampoline 2.0)

The POSIX port (cont'd)

Some technical info:

- ▶ VIPER2 is a multithreaded Python framework
 - ▶ each Trampoline communicates with an instance of the ECU class
 - ▶ each ECU has its own thread (synchro with its Trampoline)
 - ▶ each ECU is also composed of a set of devices
 - ▶ new devices are created by implementing an abstract class
 - ▶ a global scheduler synchronizes the evolution of devices in the virtual time
- ▶ The communication between VIPER and Trampoline relies on two small C libs based on POSIX signals, POSIX shm and POSIX semaphores (contribution of Fabien Juif)
- ▶ We hope that the community of Trampoline users will develop new devices (and we shall provide some support for sharing the corresponding codes)

Main characteristics of Trampoline

- ▶ ~~support for real-time multithreading~~
- ▶ ~~small memory footprint~~
- ▶ ~~support for early development stages on POSIX ...~~
- ▶ compatible with OSEK/VDX OS and AUTOSAR OS API
- ▶ portable (clear separation between hardware dependent and hardware independent code)
- ▶ work-in-progress:
 - ▶ kernel support for multicritical systems (memory and timing protection)
 - ▶ scriptable (w/ python) virtual execution environment (viper2)
 - ▶ logging and post-mortem analysis (T³: Trampoline Trace Toolkit)

Conclusion: why should you take a look at trampoline?

Trampoline is a young open-source platform for small real-time embedded systems

- ▶ alternative to FreeRTOS
- ▶ based on OSEK / AUTOSAR standards

We are looking for

- ▶ contributors: new ports, device drivers
- ▶ users: case-studies, bug reports

Thank you for your attention

Any question ?

`http://trampoline.rts-software.org/
jean-luc.bechenec@irccyn.ec-nantes.fr
sebastien.faucou@univ-nantes.fr`