Trampoline
OSEK-VDX & AUTOSAR Compliant
Open Source Real-Time Operating System

Jean-Luc Béchennec, Mikaël Briday, Sylvain Cotard,
Sébastien Faucou, Yvon Trinquet

Journée ACTRISS « Outils ». November 7, 2012
History and Motivation

2005 Start of development, **Goals:** 1) automotive industry standard compliant RTOS, 2) Open Source RTOS for teaching and research purposes.

Early 2007 Start of O4A (Open for AUTOSAR) PREDIT project, Validate OSEK compliance, Add AUTOSAR 2 compliance. Start of cooperation with See4sys.

Late 2008 Start of O4A II PREDIT Project, Add AUTOSAR 3 compliance.

Late 2010 Start of RESPECTED ANR Project, currently adding Multicore support and modular scheduler.
A static RTOS

« Static » means no object creation or deletion at runtime. Tasks, resources (mutex), events (signalization), . . . are all known at compile time.

Trampoline encompasses:

- A real-time kernel (trampoline os);
- A communication extension (trampoline com and trampoline ioc);
- An off-line kernel configuration tool (goil);
- A virtual environment to prototype applications on POSIX workstations (viper);
- A runtime verification tool (enforcer).
Architecture

Kernel
- Interrupt dispatcher
- Counter manager
- Scheduler

BSP
- Interrupt handler
- System Call handler
- Context switching
- Memory protection manager

How it works

J-L Béchennec et al. (IRCCyN)
Workflow

- Application OIL description
- GoIL v2 Compiler
- Application Description + generated code. C Source(s)
- C Compiler + Linker
- Binary executable

System Configuration Templates
Kernel C Sources
OS infrastructure C + Assembly Sources

Application C Source(s)
A word about Enforcer

How it works

Monitoring service

Event analysis

Event table

Event Handler

Monitor update

Transition table

True occurs

False occurs
CPU only_one_periodic_task {
    OS config {
        STATUS = EXTENDED;
        BUILD = TRUE {
            APP_SRC = "periodic.c";
            TRAMPOLINE_BASE_PATH = "../../..";
            APP_NAME = "periodic_exe";
        };
    };
}

APPMODE std {};

ALARM one_second {
    COUNTER = SystemCounter;
    ACTION = ACTIVATETASK { TASK = my_periodic_task; };
    AUTOSTART = TRUE { APPMODE = std; ALARM_TIME = 100; CYCLE_TIME = 100; };
};

TASK my_periodic_task {
    PRIORITY = 1;
    AUTOSTART = FALSE;
    ACTIVATION = 1;
    SCHEDULE = FULL;
};
Porting is quite easy. It is estimated to 1 week of work for someone who know the target by See4sys. It consists in:

- Writing the BSP (Interrupt handler, System Call handler, Context switching);
- Writing the templates to generate the hardware dependent task and ISR descriptors;
- Writing the OIL description of the available interrupt sources.

Currently the following targets are supported:

- POSIX
- PowerPC (32 bits ISA and VLE ISA)
- ARM (32 bits ISA, Thumb soon)
- C166
- AVR
Small Memory Footprint

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

<table>
<thead>
<tr>
<th></th>
<th>Basic task / ISR</th>
<th>Extended task</th>
<th>Alarm</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static descriptor</td>
<td>4 ptr + 4 bytes</td>
<td></td>
<td>3 ptr + 2 bytes</td>
<td>-</td>
</tr>
<tr>
<td>Dynamic descriptor</td>
<td>1 ptr + 2 bytes</td>
<td>1 ptr + 4 bytes</td>
<td>3 ptr + 4 bytes</td>
<td>2 ptr + 2 bytes</td>
</tr>
</tbody>
</table>

- Could be improved by using bitfields instead of uint8, but portability would be decreased
- Simple applications run on 8 bit CPU, RAM < 2KB, ROM < 8KB
OSEK conformance has been specified in the MODISTARC ESPRIT project. MODISTARC defines:

- the testing methodology
- the test plan
- the test procedure

We implemented a test suite that includes the MODISTARC test plan/procedure + additional tests + AUTOSAR tests = 142 tests. The test suite is packaged with Trampoline under the GPL licence.
More to come

Work in progress within ANR RESPECTED (LEAT, LAAS, IRCCyN, See4sys):

- Multicore kernel design
  - Formal verification of synchronization schemes
- Modularization:
  - plugging other schedulers;
  - plugging API.
Who is using Trampoline in Teaching?

- Université de Nantes
- Ecole Centrale de Nantes
- Université de Rennes
- INSA Toulouse
Who is using Trampoline in Research?

- Caroline Lu PhD (LAAS/Renault)
  "Robustesse du logiciel embarqué multicouche par une approche réflexive : application à l’automobile"

- Dominique Bertrand PhD (IRCCyN)
  "Contribution à la robustesse des systèmes temps-réal embarqués"

- Sylvain Cotard PhD (IRCCyN/Renault)
  "Méthode d’exploitation sûre des systèmes temps réel embarqués multicœur et multiprocesseur"

- Hélène Martorell PhD (LAAS/Renault)
  "Architectures et processus de développement permettant la mise à jour dynamique de systèmes embarqués automobiles"


Who is using Trampoline in the Industry?

- See4sys: Trampoline is included in their AUTOSAR Basic Software offer;
- PSA: As part of collaboration with See4sys;
- Renault: In various CIFRE PhD Thesis;
- Other car manufacturers (at least 2);
- Dunasys: Currently for evaluation.
To conclude

Open Source distribution and spreading

- ≃ 4000 downloads per year;
- Eases the research works;
- Eases the collaborations;
- Interesting for teaching;
- Not an obstacle for industry;
- Small community;
- Many contacts with students / academics / engineers.
Questions?