OSEK-Like Kernel Support for Engine Control Applications Under EDF Scheduling

Vincenzo Apuzzo, Alessandro Biondi and Giorgio Buttazzo

ReTiS Laboratory
Scuola Superiore Sant’Anna, Pisa, Italy
Towards the use of EDF in real-world engine control applications

Not only periodic tasks!
Engine control applications also include adaptive variable-rate tasks

Benefits in terms of schedulability have been observed (in theory) under EDF scheduling

This Work
- OSEK-like RTOS support for EDF scheduling of engine control applications
- Simulation Framework
A LOOK INTO ENGINE CONTROL APPLICATIONS
Engine control applications include

- **Periodic tasks** with fixed periods: 1 - 500 ms
- **Angular tasks**, linked to the rotation of the crankshaft
Introduction
Engine-triggered Tasks

- **Engine-triggered** tasks – *single activation per revolution*

Inter-arrival time given a fixed speed $\omega$

$$T = \frac{2\pi}{\omega}$$

**Graph**

- $T_{\text{min}} = 10 \text{ ms}$
- $T_{\text{max}} = 120 \text{ ms}$

**Values**

- $\omega_{\text{min}} = 500 \text{ rpm}$
- $\omega_{\text{max}} = 6500 \text{ rpm}$
Engine-triggered Tasks

- Engine-triggered tasks – single activation per revolution

![Diagram](image)

Inter-arrival time given a fixed speed

\[ T = \frac{2\pi}{\omega} \]

VARIABLE-RATE TASKS

\[ \omega_{\text{min}} = 500 \text{ rpm} \Rightarrow T_{\text{max}} = 120 \text{ ms} \]

\[ \omega_{\text{max}} = 6500 \text{ rpm} \Rightarrow T_{\text{min}} \approx 10 \text{ ms} \]
Engine-triggered Tasks

Suppose a fixed \textit{WCET} for the task

![Diagram showing worst-case execution time (WCET) and CPU load over time with an overload indication.](image)
Suppose a fixed **WCET** for the task

**Engine-triggered Tasks**

worst-case execution time

\[ WCET \]

TASK

\[ \text{time} \]

**CPU load**

\[ 100\% \]

**Overload**
Engine-triggered Tasks

To prevent overload at high rates, different control implementations are used.
Engine-triggered Tasks

To prevent overload at high rates, different control implementations are used.
Adaptive Variable-Rate Tasks

- The AVR task implements a number of execution modes
Scheduling Infrastructure

固定优先级调度

OSEK/AUTOSAR RTOS

定时器

发动机

CPU

ECU

Set of Periodic Tasks
$$\tau_i (C_i, T_i, D_i)$$

Set of engine-triggered tasks
$$\tau_i^* (C_i(\omega), T_i(\omega), D_i(\omega))$$
IS FIXED-PRIORITY SCHEDULING THE BEST CHOICE FOR ENGINE CONTROL APPLICATIONS?
FP Scheduling of AVR Tasks

Since the inter-arrival time vary a lot with $\omega$, any fixed priority assignment may not be optimal for some speed!
EDF Scheduling of AVR Tasks

- Job priorities are adapted at run time as a function of the engine speed at their release time.

- Variable relative deadline for each job.

- This is still EDF! (job-level fixed-priority)
Deadline Assignment

- **Engine-triggered** tasks – **Dynamic** condition

\[ T = \frac{2\pi}{\omega} \]

\[ C(\omega) \]

\[ \theta(t) \]

\[ \alpha \geq 0 \quad \alpha < 0 \]
Deadline Assignment

- **Engine-triggered** tasks – Dynamic condition

The deadline is assigned considering the *earliest possible next activation* given by the maximum acceleration.
Benefits of EDF

Experimental results from [1]

EDF is “practically” optimal

Speed-up factor analysis
Guo and Baruah [2]

Depends on engine speed and maximum acceleration

~1.1


LET'S TRY TO USE EDF FOR REAL-WORLD ENGINE CONTROL APPLICATIONS...
Motivated by the benefits of EDF observed in theory

- Design and implementation of a RTOS support for engine control applications under EDF scheduling

- Being OSEK/AUTOSAR the de-facto standard in the automotive industry

  - Minimal changes to the standard OSEK API
  - Integration with the OSEK standard configuration language (OIL)
Our Goal

Existing engine-control application

OSEK RTOS
Our Goal

Existing engine-control application

This Work

less changes as possible
ERIKA Enterprise is an OSEK/VDX certified RTOS

Offers a suitable open-source license allowing the static linking of closed source code

Typical footprint around 2-4KB Flash

Used by several automotive and white goods companies
OSEK-certified
- BCC1
- BCC2
- ECC1
- ECC2

FP
minimal impl. of fixed-priority scheduling

EDF
EDF scheduling + (M)SRP

HR
two-level hierarchical scheduling (M)BROE

FRSH
EDF-based per-task resource reservation

OSEK-like API
Impact

- RTOS should be aware of the parameters of AVR tasks and the engine
- Needed support for variable relative-deadline as a function of the engine speed
- Needed extensions at the OSEK Configuration Language (OIL)
- Needed new support for deadline buffering to manage overloads
- Different requirements for stack sharing
- ...

Activation of an AVR task

**Interrupt:**
CrankshaftAngle.ZERO

\[
\omega = \text{read_rotation_speed}();
\]

\[
\text{ActivateTask}(\text{AVRtask}, \omega);
\]

Not part of the OSEK standard API
The deadline of each job depends on the engine speed $\omega$ (at the job release time)

$D(\omega)$

engine speed $\omega$

Must be computed every time a job of an AVR task is activated
Deadline Computation

run-time overhead
footprint
error
Deadline Computation

Relying on standard C libmath

- ~450 cycles
- $5.4 \mu s$ on SMT32F4 @ 168Mhz

run-time overhead

footprint

timeout

guaranteed

error
Deadline Computation

- run-time overhead
- footprint
- error

**fastSQRT algorithm**

- ~188 cycles
- 2.2 μs on SMT32F4 @ 168Mhz
- error < 0.04%
Deadline Computation

look-up table
64/128 entries
@ 32 bit

~25 cycles
0.3 μs on SMT32F4
@ 168Mhz
error 0.2/0.05%

run-time overhead

footprint
error
Experimental Results

Run-time overhead for the *ActivateTask* context switch + deadline computation + ready queue management + ...

<table>
<thead>
<tr>
<th>Num. of Tasks</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EDF-AVR (FastSQRT) MAX</strong></td>
<td>μs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cycles</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDF-AVR (FastSQRT) AVG</td>
<td>μs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cycles</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDF-AVR (Lookup Table) MAX</td>
<td>μs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cycles</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDF-AVR (Lookup Table) AVG</td>
<td>μs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>cycles</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

STM32F4 @ 168Mhz – GNU ARM Compiler

Fixed-priority
420 cycles
2.2 μs
Experimental Results

Footprint – 2 periodic tasks + \#n AVR tasks

STM32F4 – GNU ARM Compiler with -Os
A SIMULATION FRAMEWORK
Simulation Framework

- **Lauterbach** is the world’s larger producer of hardware assisted debug tools for microprocessors

- **TRACE32® PowerView IDE**

- Lauterbach makes available a version of their IDE based on an **instruction-set simulator** → trace & debug without any hardware!
Simulation Framework

- The TRACE32 simulator offers a standard interface named *Peripheral Simulation Model (PSM)*

- The PSM allows developing *custom* simulated peripheral devices
  - react to events (e.g., memory read);
  - access to the simulated CPU registers;
  - ...

![Diagram showing Lauterbach TRACE32 connected to PSM and custom simulated peripheral device]
void myMemoryReadHandler()
{
    <…>
    PSM_TRACE_WRITE_MEM(…);
    <…>
    PSM_TRACE_WRITE_REG(…);
    <…>
}

on write
void mySimulatedInterrupt()
{
    <…>
    PSM_TRACE32_RISE_INT(3);
    <…>
}
Simulation Framework

\[ \omega, \alpha \]

Random Speed Pattern

File

Crankshaft Simulator

Simulated STM32F4

TRACE32 IDE

Free Running Timer Simulator

Lauterbach TRACE32

ERIKA Application Binary

\[ w(t) \]

\[ t \]
DEMO
Conclusions

- We presented a new RTOS support for EDF scheduling of engine control applications.

- The implementation has been conceived to require **minimal changes** to existing applications (OSEK-like API, integration with OIL).

- Run-time overhead and footprint are not problems (**+1.5 μs** and **+500** bytes over an implementation of fixed-priority scheduling).

- We also present a powerful **simulation framework** for studying the execution of real code under (but not only limited to) the proposed RTOS.
Future Work

- We are going to test this implementation with a real engine control application controlling a real engine

- Integration of the TRACE32 simulator with MATLAB Simulink and/or other physical simulation tools

soon available as open-source

http://erika.tuxfamily.org/
Acknowledgements

Thanks to Paolo Gai from Evidence S.R.L. and Maurizio Menegotto from Lauterbach Italia for their support which helped to improve this work.
Thank you!

Alessandro Biondi
alessandro.biondi@sssup.it