EVIDENCE®

EMBEDDING TECHNOLOGY

A fully Open-Source Platform for Automotive Systems

Paolo Gai,
CEO Evidence Srl

Implementation by

Bruno Morelli,
Evidence Srl

The company

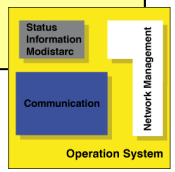
Founded in 2002, we do custom design and development of software for small embedded devices

~20 qualified people with an average age of 34 years, 30% PhD

Experience in automotive, industrial, white goods

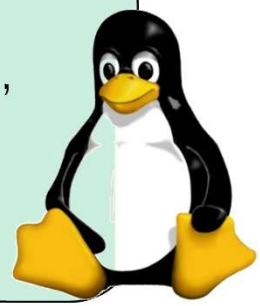
RTOS and MCU skills

- OSEK/VDX, AUTOSAR,
- Automatic code generation

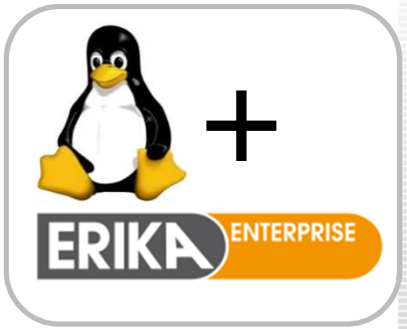


Embedded Linux skills

- 8 Yrs experience in custom BSPs, U-Boot, kernel drivers,
- Initial developers of the SCHED_DEADLINE patch



Everything in one slide



- We created a completely Open-source solution for Automotive systems
  - Linux for HMI, communication and logging
  - ERIKA Enterprise for real-time performance and control
- We will show a demo on a dual core Freescale iMX6

ERIKA ENTERPRISE http://erika.tuxfamily.org

- Free RTOS for automotive devices
- Open-source license, static linking of closed source code
- ERIKA Enterprise is the first and only OSEK/VDX certified open-source RTOS

Agenda

- IVI system requirements and multicore devices
- Main features of Erika Enterprise
- Success stories
- Towards a fully integrated Open-Source solution with Linux and Erika Enterprise
- (some) implementation details
- Demo on Freescale iMX6

Main Requirements of future IVI systems

- Fast Boot
  - there must be a subsystem ready to go in a few ms
  - Linux boot times are usually in the order of seconds
- Real-Time support
  - there must be a subsystem with real-time performance
  - e.g. CAN Bus, motor control
- Quality of Service
  - IVI applications need soft-realtime support
  - for video/audio content

Infotainment, Linux, and multicores

- Next generation infotainment systems will be multi-core
- They can host more than one OS

What about

creating a complete
open-source environment
for automotive systems integrating

Infotainment + OSEK/VDX/AUTOSAR

on the same chip?

Opportunity

Linux Embedded

- Drivers, Displays, and communication infrastructure
- Soft Real-Time support using Linux and SCHED_DEADLINE

ERIKA Enterprise

- Hard Real-Time support
- Open-source
- OSEK/VDX system, born for automotive

on a single multicore chip!!!

Something about ERIKA Enterprise

ERIKA ENTERPRISE

http://erika.tuxfamily.org

- ERIKA Enterprise is an RTOS OSEK/VDX certified
- ERIKA Enterprise implements an API inspired to a subset of the AUTOSAR API
- With a suitable open-source license allowing static linking of closed source code
- Typical footprint around 2-4KB Flash
- Configured through RT DRUID Eclipse plugins

OSEK/VDX API support

ERIKA Enterprise supports the OSEK/VDX API

Complete implementation of the following components:

- OSEK OS (BCC1, BCC2, ECC1, ECC2)
- OSEK OIL
- OSEK ORTI using Lauterbach Trace32
- OSEK COM (CCCA, CCCB)

- Additional research conformance classes implementing Earliest Deadline First and Resource Reservation (similar to the SCHED_DEADLINE patch for Linux)

Licensing

ERIKA ENTERPRISE

- License: GPL + Linking Exception
  - http://en.wikipedia.org/wiki/GPL_linking_exception
  - Proprietary applications can be statically linked with the RTOS!

RT DRUID

- License: EPL Eclipse License
  - http://en.wikipedia.org/wiki/Eclipse_Public_License

Industrial usages: Cobra AT

The first one was Cobra AT

with:

2009 – feasibility for a OEM product
based on Freescale S12XS

2012 – Cobra **Park Master product platform**

http://www.cobra-es.com/parkingaid_350.html

Magneti Marelli

Then came Magneti Marelli Powertrain Bologna

With support for:

- PPC MPC5674F (Mamba)
- MPC5668G (Fado)
- Multicore support
- AUTOSAR Memory Protection

Then…

Aprilia Motor Racing on PPC

FAAM on S12XS

esi-RISC port (made by Pebble Bay)

A white goods company

TI Stellaris Cortex M4F, Renesas 2xx and AUTOSAR-like drivers

Demo @ Freescale Automotive seminar

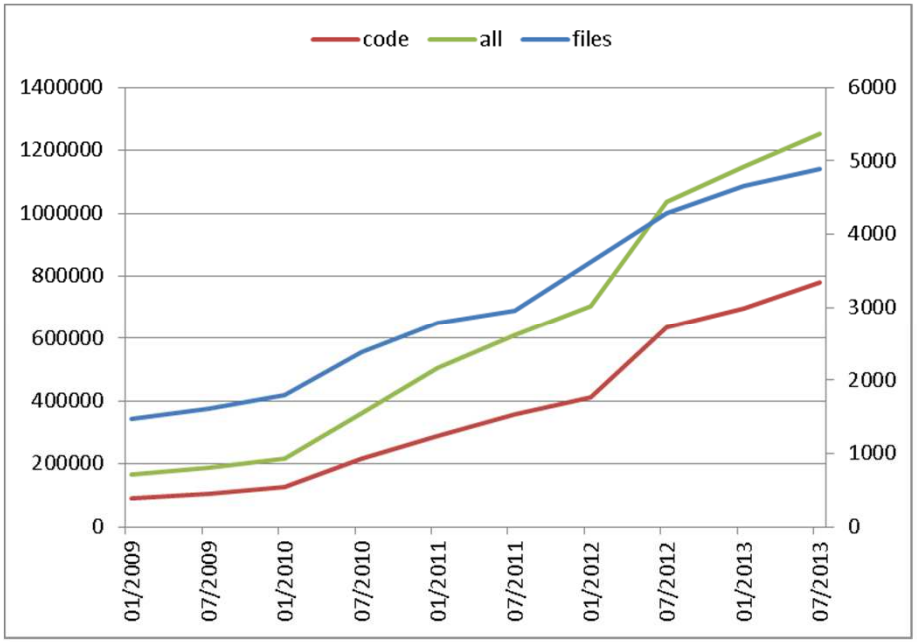
Other undisclosed companies

PPC Leopard and Infineon AURIX

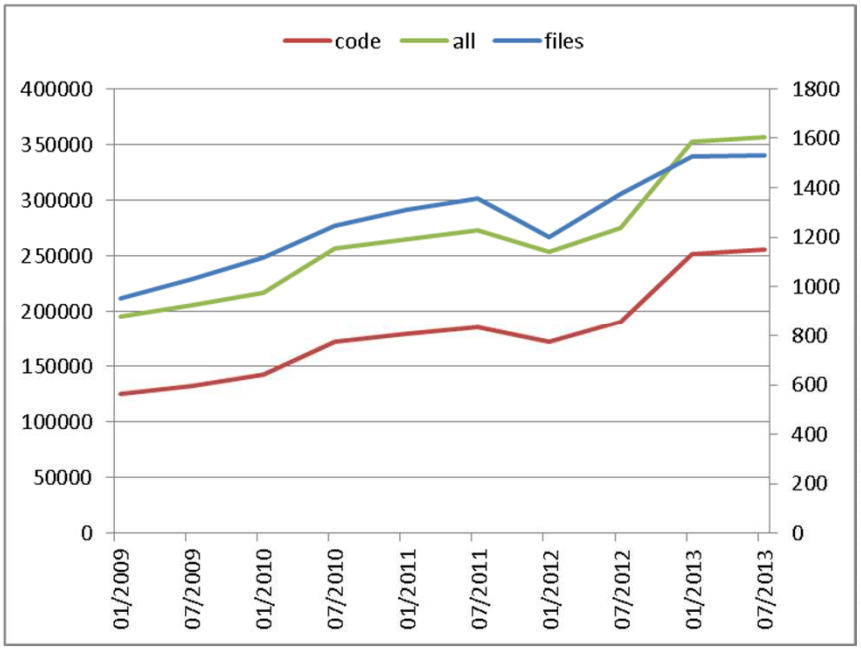
code size

The code base increased 3x from 2009 to 07/2013

ERIKA Enterprise



RT-Druid



Hardware supported

ERIKA Enterprise supports the following microcontrollers:

Microchip	PIC24, dsPIC, PIC32
Altera	Nios II
ARM	ARM7, Cortex M0, M3, M4, A
Lattice	Mico32
Freescale	S12XS, S12G
Freescale	PPC z0, z4, z6, z7 (Mamba, FADO, Leopard)
Infineon	Tricore AURIX
Atmel	AVR5, Arduino
Ensilica	esi-RISC
TI	MSP430, Stellaris Cortex M4
Renesas	R21x

A Porting guide available on the ERIKA Wiki!

Other features

Multicore support

- code partitioned in the various cores
- a copy of the RTOS for each core
- support for Lauterbach debuggers

AUTOSAR OS –like support

- Not officially part of the AUTOSAR consortium
- We are implementing AUTOSAR OS specifications
- Including Memory Protection, OS Applications, …

AUTOSAR Drivers

- MCAL + some complex drivers available for Cortex M4 and Renesas R2xx

Code testing

- A subset of ERIKA Enterprise has been checked for MISRA C compliancy with Magneti Marelli Powertrain

- Regression tests available, run daily on Jenkins
- Benchmarks available on the web site

The development community

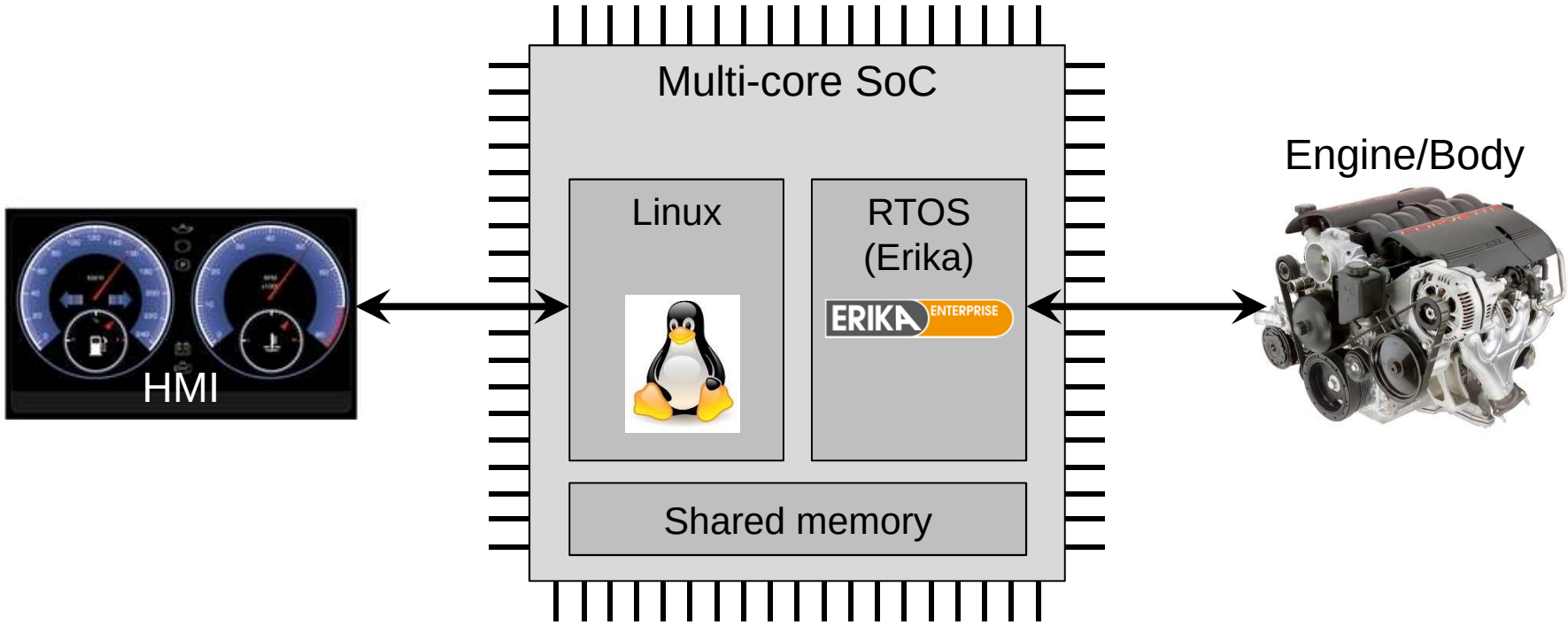
http://erika.tuxfamily.org

- SVN repository open to the public
- Wiki and forum
- Application notes
  - Template system available in RT-Druid
- libraries for
  - console
  - uWireless (802.15.4 with beaconed mode / GTS support)
  - ScicosLab
  - Motor control
  - TCP/IP
  - CMOS Cameras, tracking
  - USB
  - various sensors
  - ball & plate, inverted pendulums, robot swarms

ERIKA Enterprise OSEK/VDX
+
Linux
Multicore Integration

Towards a fully Open-Source platform

We envision the possibility to exploit multi-cores to run Linux and Erika Enterprise complementing each other!



Three scenarios

1) Linux boots, ERIKA = special «device» for Linux

- slow! → ERIKA needs to wait for Linux boot

2) Hypervisor-like approach

- both ERIKA and Linux as hypervisor «clients»

3) ERIKA boots from U-Boot

- modified U-Boot to boot both ERIKA and Linux

Our current choice

Demo based on a Freescale iMX6

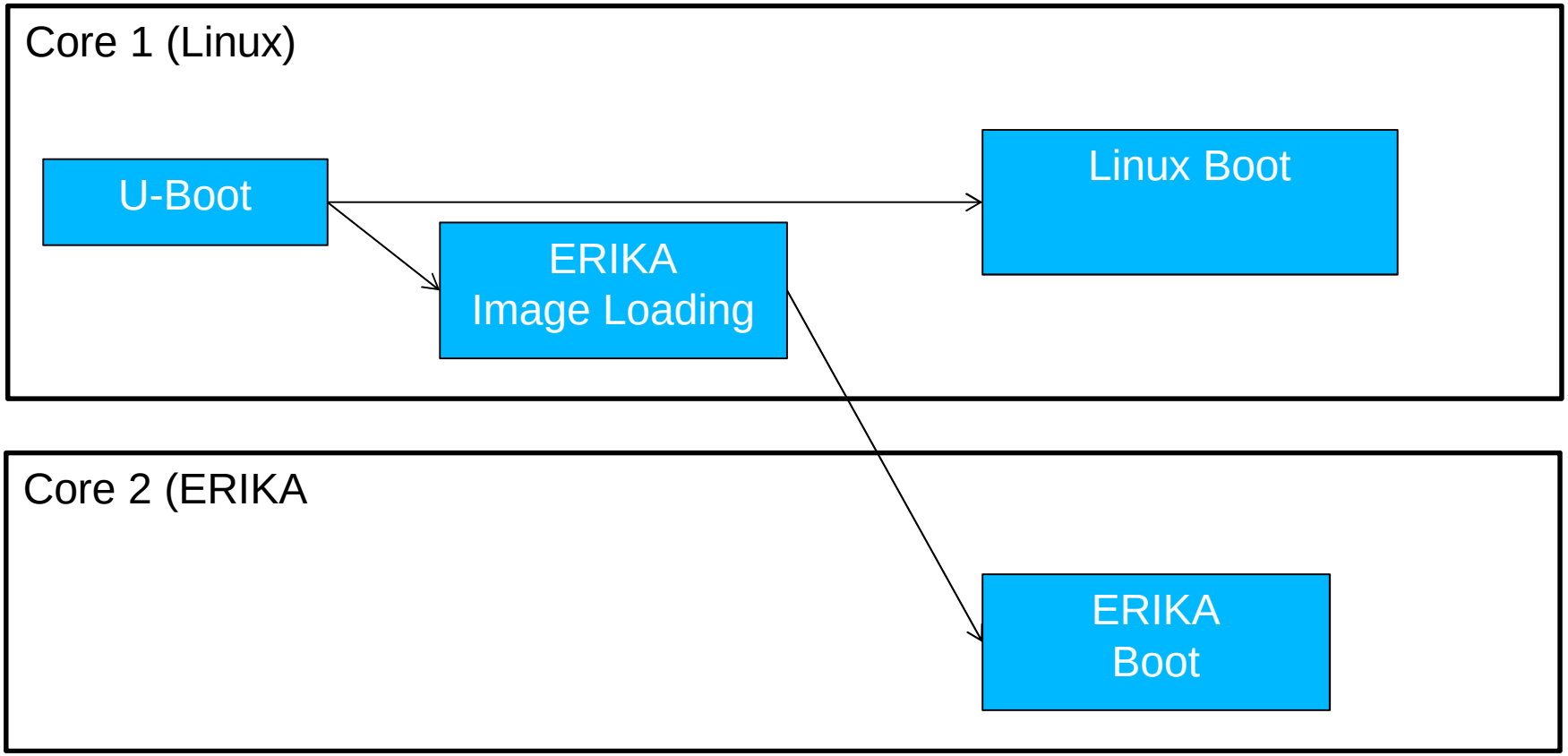
We let U-Boot handle the multicore boot

- ERIKA starts almost immediately
- Linux can start afterwards

No hypervisor

- could be useful in some cases to protect the behavior of misbehaving applications
- limited need because we statically allocate a CPU to each OS

The idea…



Interaction model

Linux → ERIKA

- Linux can trigger the following actions:
  - activate a task
  - set an event
  - start an Alarm
  - increment a counter

  (similar to those doable on a remote core of an AUTOSAR OS)
- Linux can stop and reload the ERIKA application

Linux ←→ ERIKA

- Simple asynchronous message passing allowing asynchronous read/write of variable length buffers on predefined channels

The demo…

- Implemented on a iMX6 board from Engicam (http://www.engicam.com/prodotti/icorem6.html)
- U-Boot loads ERIKA, then Linux
- ERIKA generates a SawTooth signal
- Linux reads the message and displays the data
- A slider can be used to set the sawtooth signal amplitude
  - implemented through messages
- Simulated LED
  - implemented through interprocessor interrupt
  - there can't be a demo without a Blinking Led!

U-Boot code changes

we added the `cpu` command to U-Boot

- (cherry pick from PPC to iMX6)

Multiprocessor CPU boot manipulation and release

`cpu <num> reset`

← Halts cpu <num>

`cpu <num> status`

← prints latest `<addr>` and `r0`, plus the status

`cpu <num> release <addr> [args]`

← Restart of cpu <num> at <addr> with a value for the r0 register

Linux code changes

- Linux runs on a subset of the available CPUs
  - 1 CPU dedicated to ERIKA
- IRQs are mapped statically to cores
  - additional boot parameter to map the GIC IRQs that Linux cannot use

```
git_skip_intid=142-147,152,180,205-220
```

Linux code changes

- a fixed amount of memory is allocated to ERIKA
  - ERIKA allocated in the first part of the RAM, Linux afterwards

- Idle time does not change CPU frequency
  - Linux by defaults reduces the CPU frequency on idle time

ERIKA code changes

- ERIKA is statically linked on the first 128 Mb of the available RAM
- the Memory Protection Unit (MPU) has been programmed to limit the possibility to write only inside the allocated memory
  - it will not destroy Linux!
- the OIL file used to configure ERIKA has been extended
  - Cortex A support
  - ORTI support through LAUTERBACH DEVELOPMENT TOOLS
    - We can make an AMP configuration for debugging Linux on one core and ERIKA (with ORTI support) on the second core

OIL file extensions

```
CPU mySystem {
  OS myOs {
    MASTER_CPU = "master";      ← the ERIKA CPU

    CPU_DATA = CORTEX_AX {
      CPU_CLOCK = 660.0;
      APP_SRC = "main.c";
      COMPILER_TYPE = GNU;
      MODEL = A9;
      ID = "master";            ← the ERIKA CPU
    };

    CPU_DATA = LINUX;           ← the Linux CPU
```

OIL extensions

```
MCU_DATA = FREESCALE {          ← The MCU information
  MODEL = IMX6Q;
};
BOARD_DATA = ENGICAM_ICOREM6;

REMOTENOTIFICATION = USE_RPC; ← configuring RPC
USEREMOTETASK = ALWAYS;
USEREMOTEEVENT = ALWAYS;

ORTI_SECTIONS = ALL;    ← ORTI support through
                          Lauterbach Trace32
```

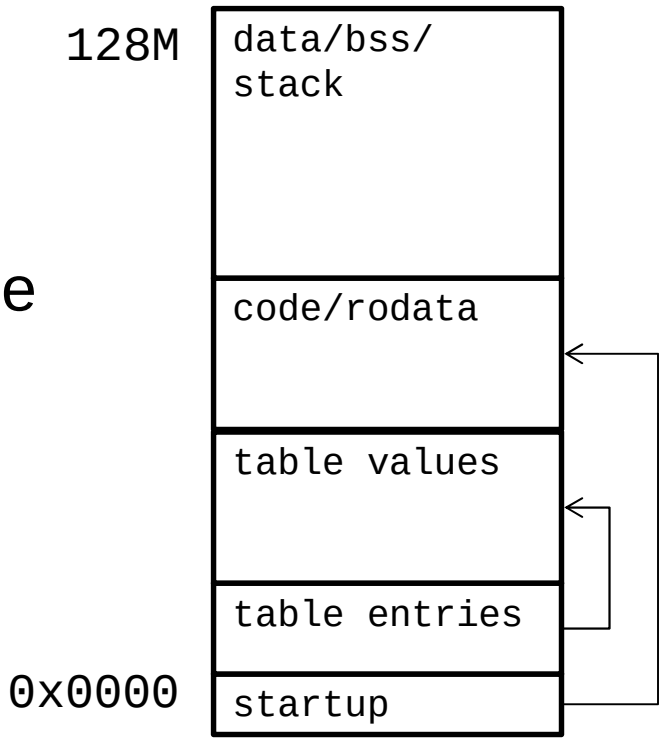
OIL extensions

```
    MESSAGE_BOX = TRUE {    ← Asynchronous message box
      NAME = "led_status";
      DIRECTION = OUT;
      MAX_MESSAGES = 5;
      MAX_MESSAGE_SIZE = 8;
    };
  };

  TASK Blinking_led_task {
    CPU_ID = "master";      ← Task mapping to the master CPU
    [...]
  };
};
```

ERIKA binary format

- we defined a custom binary format for the ERIKA images
- symbol table with a DB of the entities defined in the OIL configuration file
- a customized Linux driver reads the DB and publishes the data into `/sys` and `/dev`



Linux driver

a custom driver allows Linux to do the following actions:

- activate a task
- set an event
- start an Alarm
- increment a counter

These are remapped to interprocessor interrupts in a way similar to what specified by multicore AUTOSAR

In addition we implemented a simple asynchronous message passing primitive allowing asynchronous read/write of variable length buffers on predefined channels

/sys filesystem structure

```
/sys/class/
    EE_alarms
        [...]                ← information about alarms
    EE_buffers
        led_status           ← asynchronous message channel
            direction
            size
            [...]
    EE_tasks
        led_task             ← tasks defined in the OIL file
            activate         ← writing to this file remotely activates a task
        [...]
    mem_ex
        symbols              ← symbol table as read in the binary image
```

/sys/class/mem_ex/symbols

```
# cat /sys/class/mem_ex/symbols
                    rpc 0xb9b0003c   RPC     28
    Blinking_led_task 0xa9000000   TASK     0
       Saw_tooth_task 0xa9000000   TASK     1
    Activate_led_task 0xa9000000   TASK     2
          AlarmMaster 0xa9000000   ALARM    0
        CounterMaster 0xa9000000   COUNTER  0
           led_status 0xb9b00058   IN      40
       saw_tooth_data 0xb9b0006c   IN     400
  saw_tooth_data_max 0xb9b00080   OUT     16
```

/dev filesystem structure

```
/dev/
    mem_ex          ← ERIKA image write
    led_status      ← asynchronous message channel
    [...]
```

- it is possible to reprogram and restart the ERIKA application by writing on `/dev/mem_ex`
- asynchronous channels are inserted in the `/dev` filesystem automatically
  - you can read/write single messages
  - no remote notification – completely asynchronous

memory protection

- each core has its own Memory protection unit

ERIKA Enterprise

- single table with 1Mb pages
- ERIKA cannot write outside its own memory space
- currently we allocate 128 Mb (should be enough ☺)

Linux

- first available address after the end of the ERIKA image
- Linux can access ERIKA memory only through the driver

Other features

Spin Locks

- ERIKA and Linux use spin locks to guarantee mutual exclusion during the access to shared data structures
- the spin lock location resides in the ERIKA memory space

Interprocessor interrupt

- currently used Linux → ERIKA
  to implement remote notifications
- data exchange is implemented using asynchronous messages

Future Work

There are various lines of development possible

- Deeper integration with Linux
- Transparent integration with the GIC interrupt controller
- Porting to other microcontrollers
- Integration of the memory protection features to provide Memory protection inside the ERIKA applications

- Integration of the SCHED_DEADLINE patch on the Linux side to provide QoS support

References

- Website: http://erika.tuxfamily.org
- Wiki: http://erika.tuxfamily.org/wiki/
- Forum: http://erika.tuxfamily.org/forum/
- Subversion repositories:
  - Erika Enterprise: svn://svn.tuxfamily.org/svnroot/erika/erikae/repos/ee/trunk/ee
  - RT-Druid: svn://svn.tuxfamily.org/svnroot/erika/erikae/repos/rtdruid/trunk/rtdruid

Questions ?

Contacts

Evidence Srl

http://www.evidence.eu.com

info@evidence.eu.com