

# Qt on eT-Kernel

## Blending Qt with RTOS for future smart devices

[gondou@esol.co.jp](mailto:gondou@esol.co.jp)

Masaki Gondo, CTO, eSOL Co., Ltd.

Qt Developer Days 2011

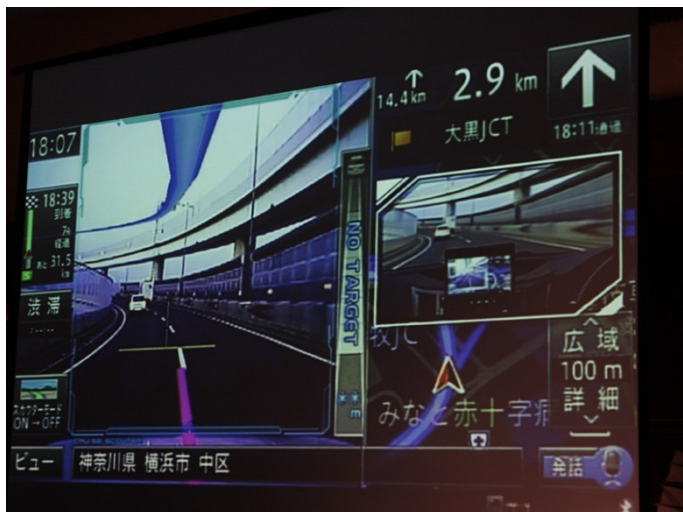
# Agenda

- Landscape of smart devices
- Blending technology options and pros-cons
- A Linux compatible multicore RTOS
- Inter-working of Qt and real-time application
- Tooling for system analysis
- Summary

# Being 'smart'

- iPhone innovated 'smart'-phones by integrating various sensors and intuitive user interface
- Context information and network information blended together
- These real-time information blended with archived information are the way to enhanced intelligence for 'smart'-devices

Examples of Augmented Reality (AR) applications



# ‘Real-time’ brings quality to smart devices

## ■ Example: music instrument application

- Android is not a great fit since its real-time capability is much worse than that of iPhone – this is one reason you do not find such good application on Android
- Making such application even more real requires more real-time processing capability



“Pianist Pro” – best iPad App. running on iOS



“CP1” by Yamaha – running on T-Kernel RTOS

# Need to think about other stuff

## ■ Widening spectrum of device grades

- The market is expanding, from high-ends, middle, low-ends, with a surge in low to mid range in emerging economies
- The price is one of key factors, but can not be a differentiator
- True architectural reuse of software (partly hardware) through product-line or alike is required for added value with competing cost performance
- Linux can be a good fit for a high-end but may not be for a differentiating low-end (e.g. Samsung's kernel configurable 'Bada')

## ■ Safety - reliability

- Mostly through the mentioned intelligence, the statistically speaking unsafe mobility 'car' can be made increasingly safer
- Reliability, though not a sufficient condition, but is a prerequisite

## ■ The last but not least – Multicore

- Legacy single-core based software onto multicore, still maintaining lower-end single-core

# Challenges defined

- Intuitive user interface and networking
  - Qt is the only cross platform framework offering exactly these
- ‘Real-time and reliable’ for intelligence and safety
  - See how iPhone gained intelligence through blending in sensors
  - Unlike my smart-phone, it has to be reliable and real-time
- ‘Scalable’ for differentiating, yet cost effective product lines
  - The high-end and possibly middle range will have Qt mixed with real-time processing, but low-end may not be capable of running full Qt
  - But from low to high, they still require real-time processing for differentiation – this portion needs to be ‘reusable’
  - Multi-core challenge previously mentioned need also be addressed

*Challenge: Blending of both media rich and real-time processing, in a scalable manner, including multi-core*

# Blending technology options and pros-cons

## ■ Blending

- Real-time/reliable processing to be blended with media-rich processing to achieve an intelligent device, with capability to support multi-core
- Though semiconductor trend is definitely multi-core, but still need to consider single core for cost-performance and reusability

## ■ Technology options

1. Linux standalone
2. Linux and RTOS - Dual-OS with or without a hyper-visor
3. Typical RTOS standalone
4. A Linux compatible multicore RTOS

# 1. Linux standalone

## ■ Pros

- Certainly Qt, Android and other application platforms run on Linux to meet ‘Media-ready’ requirement
- Open pseudo-standard, a large number of developers world-wide
  - ▶ Becoming more and more POSIX (Linux community has chosen POSIX for its standardization, has been reflected in the POSIX latest standard), like NPTL to begin with
- Multi-core ready (except for real-time scheduling)

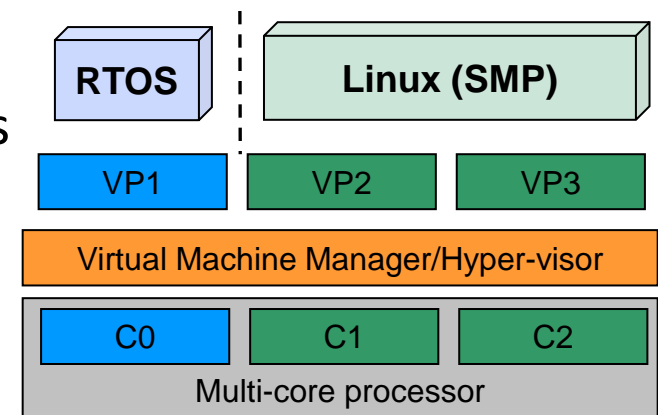
## ■ Cons

- Lack of Real-time/reliability
  - ▶ Originally designed to be desktop OS and also good for enterprise servers – still that is where the active development is
  - ▶ Very large code base – difficult to logically assure quality from its design
- Lack of scalability to the lower-end of spectrum
  - ▶ It can be run, but performance will suffer, since it is just not designed to be ‘that’ scalable – difficult to assure real-time processing

## 2. Linux and RTOS - Dual OS with or without hyper-visor

### ■ Pros

- Real-time/reliability can be achieved on RTOS side, while multi-media can be served on Linux side
- Adding a hyper-visor underneath to have two OS run in separate partitions will further protect the reliability of RTOS side from Linux side



### ■ Cons

- Inter-OS integration is limited
  - ▶ To further enhance the intelligence of system, the two worlds (real-time and multi-media) will need tighter, flexible inter-operation mechanisms
  - ▶ This is actually deviating further to the architectural principle of AMP, where “functional separation” is the root concept, thus intrinsically unfit
- Does not work well with a single core
  - ▶ Hyper-visor on a single core with two guest OS adds more significant overhead for the two virtual processor switches

## 3. Typical RTOS standalone

### ■ Pros

- Assurance of real-time processing
  - ▶ Often originally architected for embedded systems
- Assurance of reliability is easier with a much smaller kernel than Linux

### ■ Cons

- Software reuse is often limited as API being often proprietary
  - ▶ Especially for that of open source, since most are Linux based
- Often not scalable to higher end systems
- SDK for application development is rarely available
- Multi-media is not typically available
- Multi-core support is sometimes limited

## 4. Linux compatible multicore RTOS

### ■ Pros

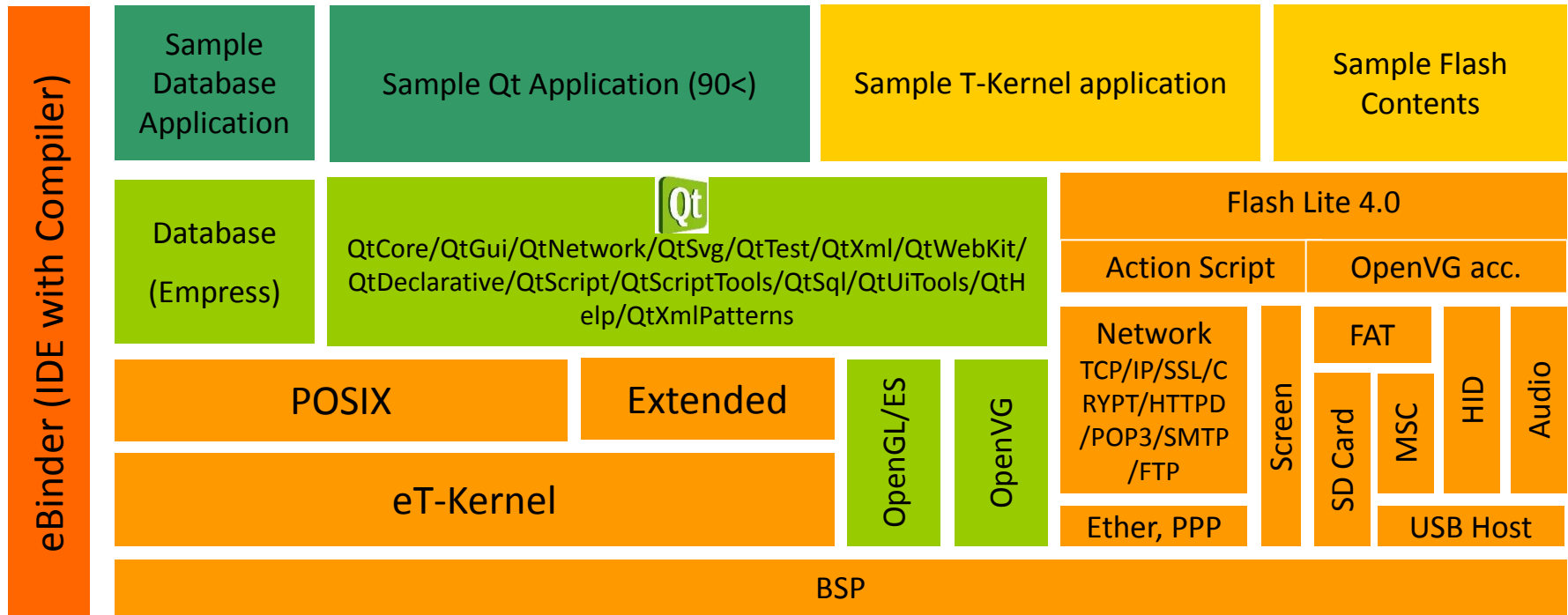
- RTOS based thus real-time capable and reliable, but also offers Linux compatible API (POSIX+) and model for multi-media ready
- The high Linux compatibility allow them to come with open application framework and SDK, such as Nokia's Qt and Android natively run on the RTOS
- Fully capable of multi-core with unique AMP blended SMP model
- Seamless integration between real-time and media-rich worlds, due to the single OS model
- The small kernel offers much better scalability with sometimes with a layered, multi-profile model, allowing re-use of real-time application from low-end to high-ends
- Multi-core partitioning is available to add further reliability

### ■ Cons

- Limited offerings with not all providing mentioned properties
  - ▶ eSOL, ENEA, QNX
- Just not so as popular as Linux - may indicate unknown risks

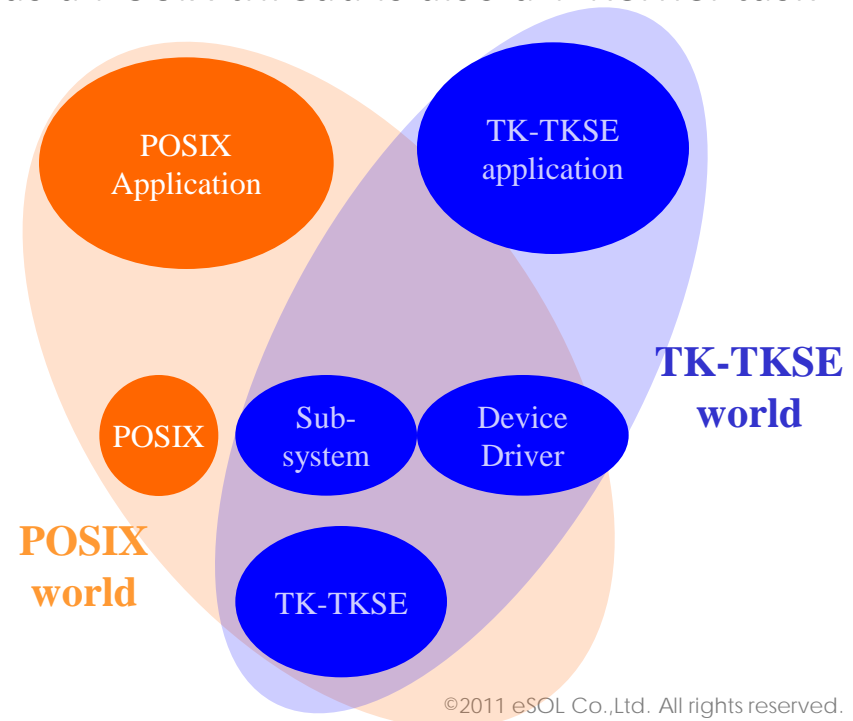
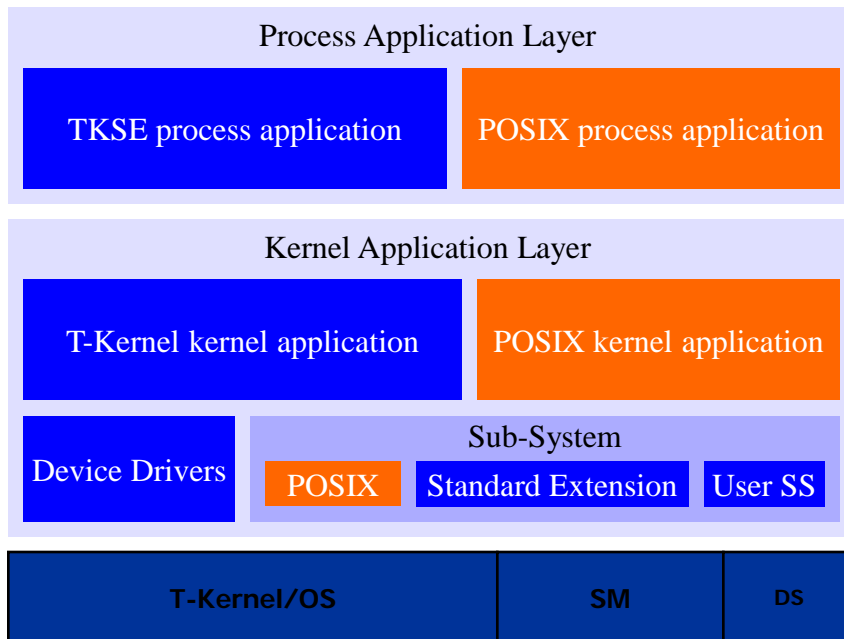
# eT-Kernel SDK

- Ready to use Qt based platform on eSOL's eT-Kernel RTOS
- Blending of real-time application and Qt application can be achieved



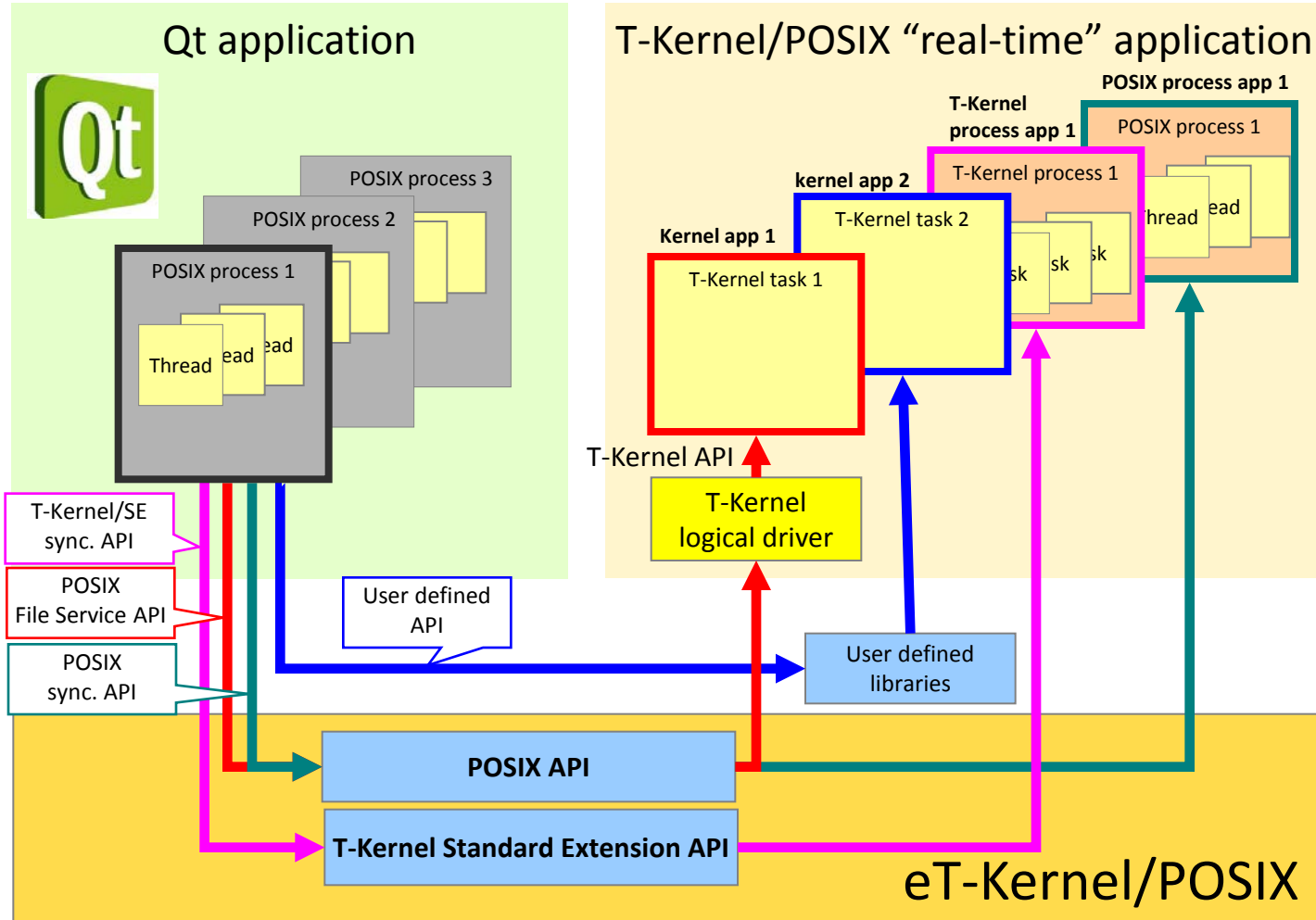
# Inter-working of Qt and Real-time application(1/2)<sup>13</sup>

- Another blend scheduling
- eT-Kernel has a kernel space application layer in addition to process space
  - Both POSIX and T-Kernel (TRON) APIs are available, essentially the same with the user space (process) APIs
  - TRON/T-Kernel is widely used RTOS API and excellent for real-time processing
- POSIX thread/process extends T-Kernel task/process
  - The two worlds can talk to each other, as a POSIX thread is also a T-Kernel task



# Inter-working of Qt and Real-time application(2/2)<sup>14</sup>

- A single OS model allows seamless integration

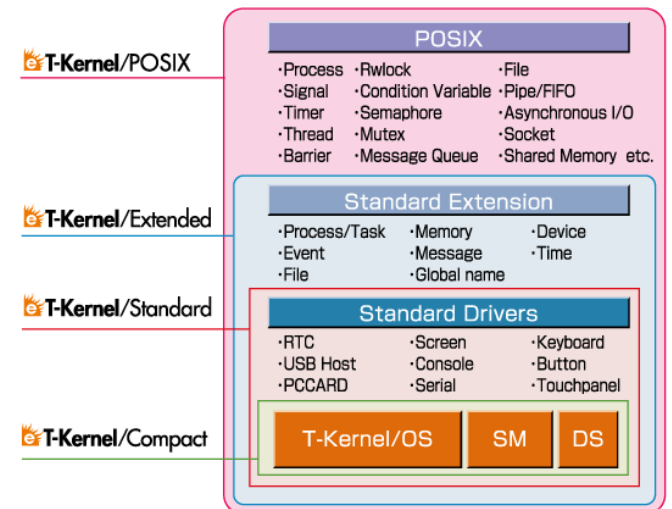
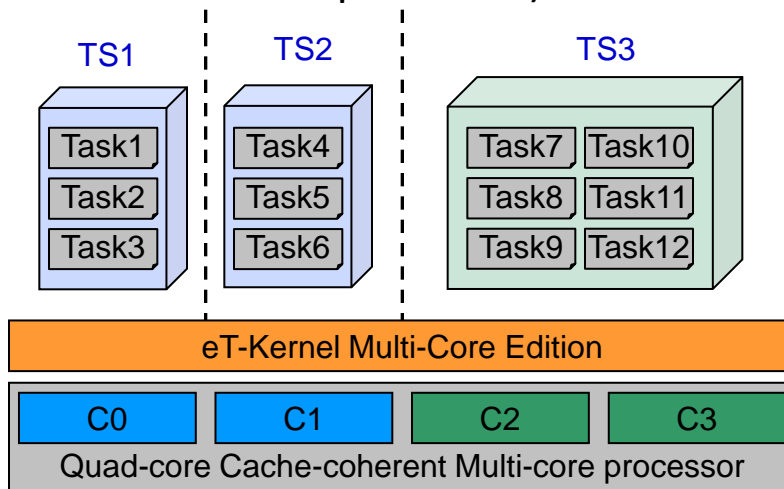


*New RTOS enables increased levels of real-time control and software flexibility for ARM MPCore multiprocessor designs*

TOKYO, JAPAN AND CAMBRIDGE, UK – Sept. 7, 2006 – eSOL Co., Ltd. and ARM [(LSE: ARM; (Nasdaq: ARMHY)], today announced the eT-Kernel Multi-Core Edition real-time operating system (RTOS) coupled with eBinder integrated tools for enhanced multiprocessor solutions, which

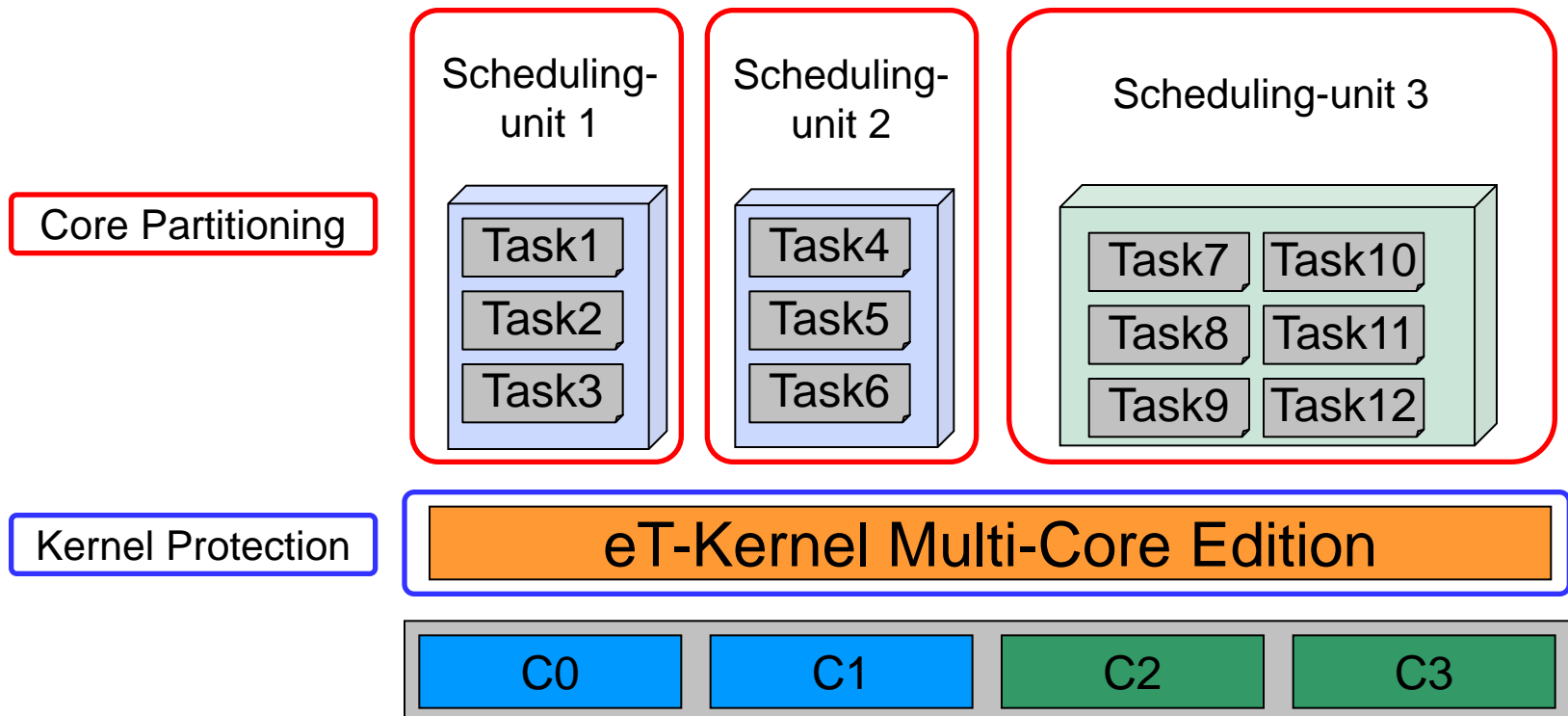
# Blended scheduling

- An unique SMP-OS with 4 modes of thread-CPU-affinity scheduling
- Single Processor Mode (SPM)
  - The most CPU-affine scheduling is essentially an AMP mode, with the programmability of SMP transparent API access
  - No thread migration in/out of the core, assuring real-time processing, together with separate kernel data structure and multi-level kernel internal locking mechanism
- The other 3 modes
  - True SMP Mode (normal SMP scheduling), SPM on TSM (normal thread-CPU-affinity), and SRL on TSM (Serialized threads within in a same process, but not those of other processes)



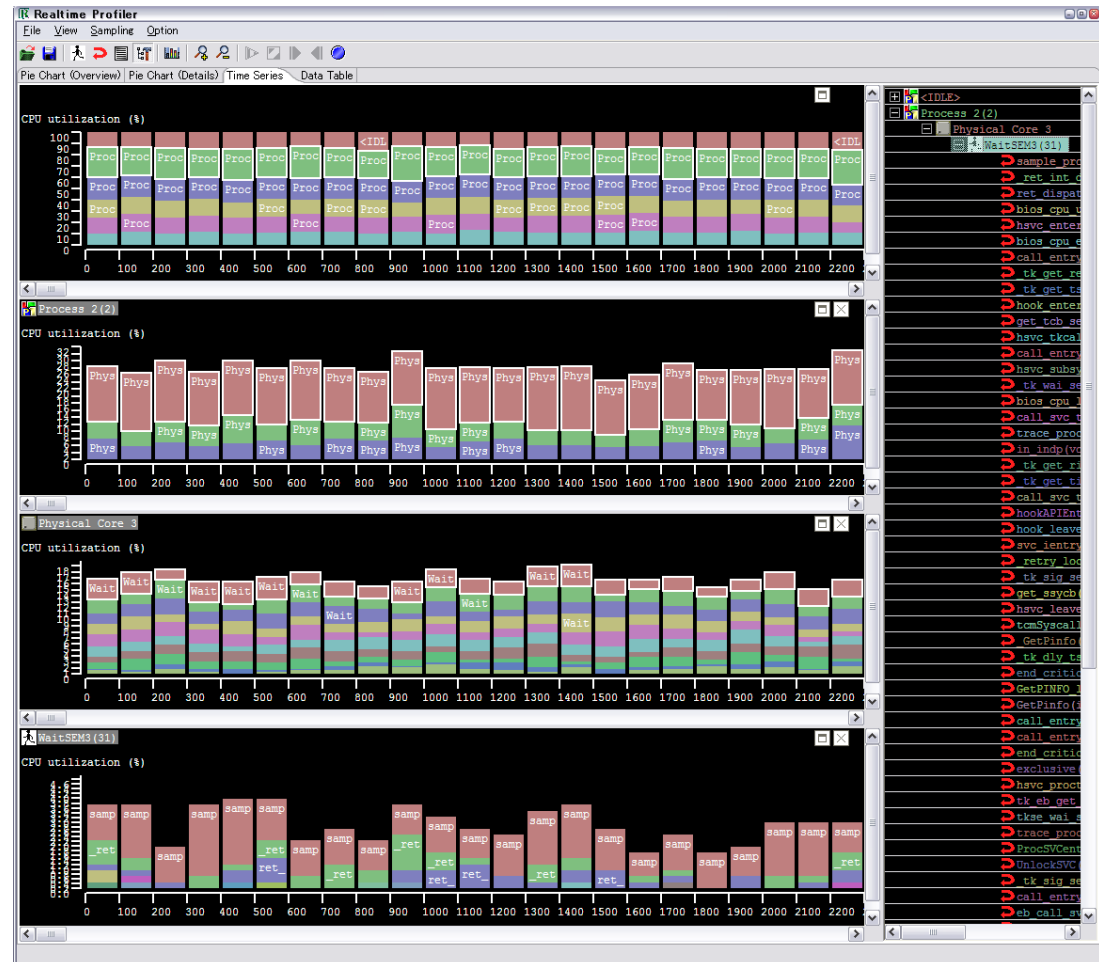
# Optional partitioning

- Example from eSOL's eT-Kernel Multi-Core Edition Memory Partitioning
- 'Kernel Protection' Protects kernel from kernel application like drivers, by running them in a mode called 'system mode', in addition to the regular kernel mode and user mode
- 'Core Partitioning' protects the kernel applications in different partitions, by blocking off the memory access



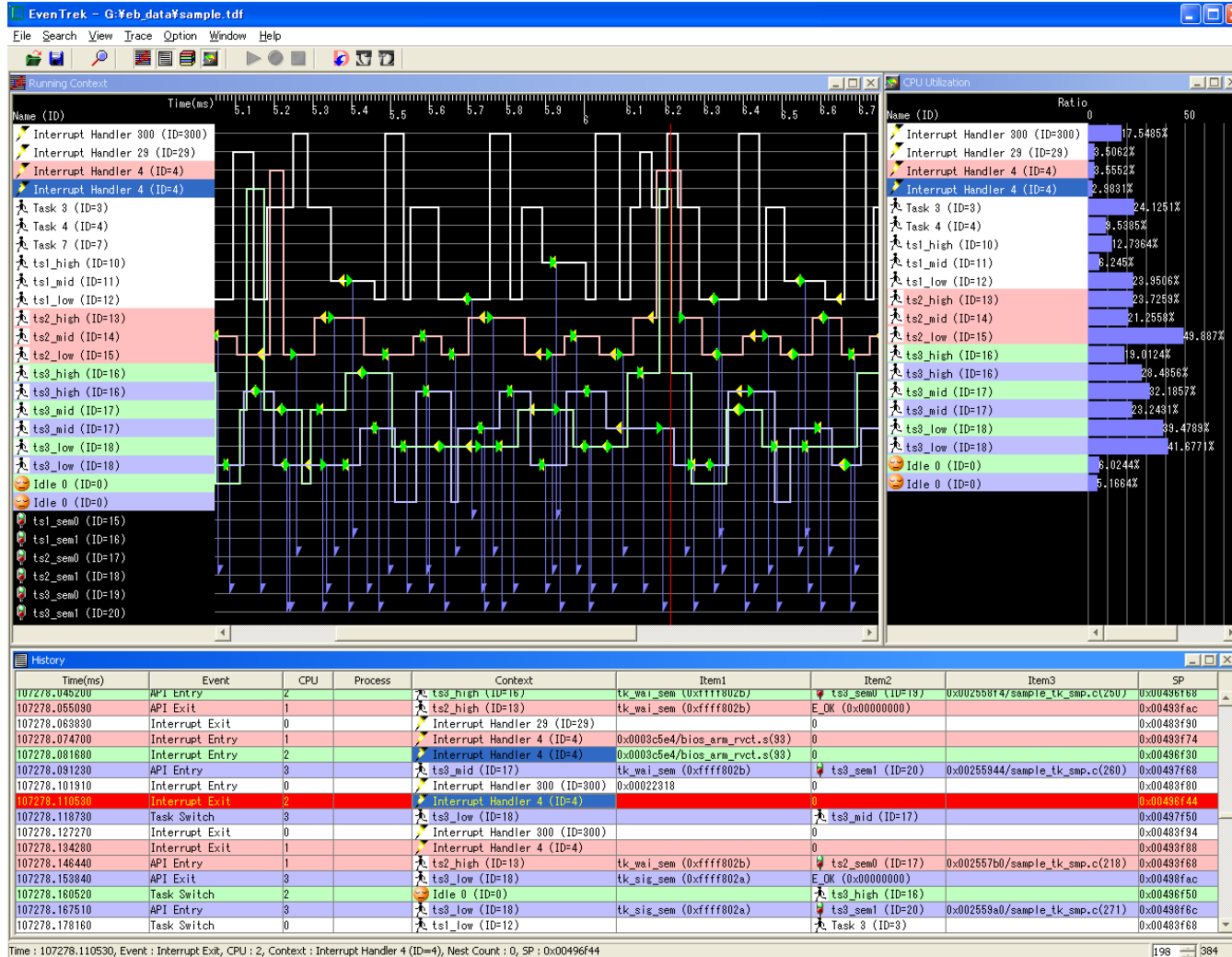
# Multicore system analysis (1/2)

## Profiler to look at the system at glance



# Multicore system analysis (2/2)

## System event trace tool to see the exact sequence



## Summary

*Challenge: Blending of both media rich and real-time/reliable worlds, in a scalable manner, including multi-core*



- 'Media-rich' are inherent of Qt
- 'Real-time and reliable' can be achieved by employing a multicore Linux compatible RTOS, allowing Qt to run natively atop itself
- 'Scalable' can be achieved by the scalable RTOS, that can allow re-usability of real-time kernel applications over wide-range of processor performance range, for differentiating, yet cost effective product lines
- The RTOS is more than capable with multi-core

# References

- “Blending Asymmetric and Symmetric Multiprocessing with a Single OS on ARM11 MPCore”
  - ARM IQ Magazine, Volume 5, Number 4, 2007
  - [http://www.esol.com/embedded/download/pdf/whitepaper\\_multicore.pdf](http://www.esol.com/embedded/download/pdf/whitepaper_multicore.pdf)
  
- “Apart Yet Blended – eT-Kernel Multi-Core Edition Memory Partitioning”
  - ARM IQ Magazine, Volume 7, Number 4, 2008
  - [http://www.esol.com/embedded/download/pdf/whitepaper\\_et\\_kernel\\_mce\\_mpo.pdf](http://www.esol.com/embedded/download/pdf/whitepaper_et_kernel_mce_mpo.pdf)
  
- “Android on a Highly Reliable Real-Time OS”
  - ARM IQ Magazine, Volume 9, Number 3, 2010
  - [http://www.esol.com/embedded/download/pdf/whitepaper\\_esol\\_for\\_an\\_droid.pdf](http://www.esol.com/embedded/download/pdf/whitepaper_esol_for_an_droid.pdf)