High-Rate Task Scheduling within AUTOSAR
Vector Addresses the Challenges of High-Rate Application Task Scheduling within an AUTOSAR System

The electrical and electronic (E&E) functions inside a modern vehicle have grown both in number and complexity; this new level of complexity has led automotive vehicle manufacturers and their suppliers to form the AUTOSAR partnership, to define a standardised yet feature-rich software architecture for within the vehicle Electronic Control Units. A common misconception is that high-rate application tasks cannot be scheduled within an AUTOSAR system. This article explains the mechanisms in place within the AUTOSAR operating system to handle the application scheduling requirements and how a successful configuration of the OS allows the software engineer to continue running the high-rate task schedules within an AUTOSAR system.

The Drivers for AUTOSAR
In recent years, the electrical and electronic (E&E) functions inside the vehicle have grown exponentially, both in number and complexity. This trend has been driven by a number of factors within the automotive industry, including an increased demand for E&E-based end-customer features, the advent and proliferation of ADAS systems and increasingly demanding legislation in diagnosing E&E system faults and Electronic Control Unit (ECU) failures. The newest vehicle functions have reached such a level of complexity that traditional automotive ECU capabilities and supplier development methods can no longer fulfil the demands imposed by the modern vehicle E&E architecture. In 2003, this increasing level of complexity led automotive vehicle manufacturers and their suppliers to form the AUTomotive Open System ARchitecture (AUTOSAR) development partnership, where their mutual goal has been to define a common and standardised software architecture for use within the vehicle ECUs, that would ultimately provide the software features, mechanisms and support necessary to meet the higher demands of the vehicle’s underlying E&E architecture. Before AUTOSAR was defined, the primary focus of the vehicle manufacturers for an ECU’s embedded software was often limited to the network communication stack and diagnostics. But with AUTOSAR, the underlying software within an ECU has undergone significant expansion; AUTOSAR “Basic Software” covers not only areas such as network communications for CAN, LIN, FlexRay, Ethernet and diagnostics protocols, but also now includes system
and watchdog initialisation and state-handling, a non-volatile memory software stack for various memory devices, an abstraction driver layer for the microcontroller peripherals and also the operating system (OS).

Increased Processing Demand from the AUTOSAR “Sweet Shop”

The AUTOSAR standards are still growing and evolving to meet the very latest vehicle E&E requirements – at the end of 2014 the AUTOSAR Basic Software attained its highest ever level of complexity so far, when version 4.2 was released. Over 90 scalable embedded software modules are defined and are supplied along with powerful configuration tools and code generators in conjunction with a set of further standards for the workflows, methodologies and data exchange formats.

When an ECU supplier chooses to equip their ECU with AUTOSAR Basic Software, they find themselves being offered a new range of feature-rich embedded software modules that become available as off-the-shelf commodities. Lengthy software development lead-times are no longer a constraint in deciding whether to build further features into the ECU, since including an additional AUTOSAR module simply becomes a tick-box within the Basic Software configuration. Software architects can find themselves like “kids in a sweet shop” with over 90 software modules to pick and choose from.

Of course, each of these software modules comes at its own cost; not only in terms of a license fee that is used to recoup the development efforts of designing and creating each module, but also in terms of processor load. Each software module will take up precious RAM and ROM resources within the microcontroller and also has functions that need to be scheduled and called, that consume valuable processor runtime overhead.

The AUTOSAR Operating System

In particular, the AUTOSAR OS offers many more features that were currently unavailable or not even considered by the ECU supplier. In the past, a simple single-tick or super-loop scheduler may have been adequate to host all the ECU application tasks. Now, with the alternative inclusion of an AUTOSAR OS, a new set of operating system features becomes immediately available to the software engineers.

The AUTOSAR OS is offered in a range of “Scalability Classes”, extending from SC1 to SC4:

- AUTOSAR OS Scalability Class 1 extends the OSEK/VDX standard operating with the addition of schedule tables
- AUTOSAR OS Scalability Class 2 extends AUTOSAR OS SC1 with the addition of timing protection
- AUTOSAR OS Scalability Class 3 extends AUTOSAR OS SC1 with the addition of memory protection
- AUTOSAR OS Scalability Class 4 extends AUTOSAR OS SC1 with the addition of both timing and memory protection

The additional features within the AUTOSAR OS do not come without a resource penalty and, like every other module within the AUTOSAR Basic Software, the increased functionality requires RAM and ROM resources within the microcontroller and consumes additional processor runtime overhead. These additional requirements would generally need to be supported with higher ECU processing power and greater microcontroller memory capacity.

A common misconception has subsequently arisen, that adopting an AUTOSAR software architecture will automatically require the ECU supplier to upgrade the internal microcontroller from a 16-bit to a 32-bit processor. Of course, this should not be the natural conclusion, since replacing an existing OSEK OS with an AUTOSAR OS SC1 would incur only a minimal additional overhead, and only if the schedule tables feature is utilised.

![Figure 1: AUTOSAR OS Scalability Classes](image-url)
Cat2 interrupts are normally used to schedule all the tasks of the Basic Software and application level software components, where the AUTOSAR modules are specified to use the AUTOSAR OS and API (application programming interface). Where the application has a specific high-rate task scheduling requirement, the software engineer can make use of the AUTOSAR Cat1 interrupt, and hook the application task directly into the microcontroller’s interrupt table using the configuration tool. The Cat1 interrupts are essentially “invisible” to the rest of the AUTOSAR system.

Cat1 interrupts offer a powerful way to interact with the hardware directly, and therefore a degree of care and diligence must be exercised when utilising this type of interrupt; the main advantage is that the latency of the interrupt is very low, and hence the required high-rate task scheduling can be achieved. However the normal protections offered by the AUTOSAR OS are not present, and therefore the interactions from within the Cat1 interrupt with the rest of the system have to be carefully considered prior to implementation.

Another aspect that has to be thought out is the allocation of the interrupt priorities across the system. The AUTOSAR OS is a fully pre-emptive operating system and therefore the AUTOSAR application tasks may interrupt each other. But it would be unacceptable for an AUTOSAR task to be allowed to interrupt, or block, the high-rate scheduled task and therefore the priorities need to be allocated such that the high-rate task has the highest priority in the system.

This means that the high-rate task can interrupt any other task in the system, including the AUTOSAR OS, and every other interrupt in the AUTOSAR system must be at least one priority level below that of the high-rate task, this is termed the “system interrupt level”.

Scheduling a High-Rate Application Task
Just like the misconception that an AUTOSAR software architecture inherently requires a 32-bit processor, a similar misunderstanding occurs that high-rate application tasks cannot be scheduled within an AUTOSAR system. Many typical automotive applications, especially within the powertrain domain, require tasks that can run at rates of every 100 microseconds (10 kHz) or even faster. Such a challenging scheduling requirement puts a very high demand on the AUTOSAR OS, and in turn on the microcontroller processing load.

For these purposes, the AUTOSAR OS provides two types of interrupt “category”:

- Cat1 interrupts are “unsupported” by the AUTOSAR OS. This means that the code to get safely into and out of the interrupt handler is not generated by the AUTOSAR OS and has to be generated in another way. [1]
- Cat2 interrupts are supported by the OS. This means that the OS’s code generator and libraries abstract the interrupt from the hardware in a portable way. [1]
Conclusions
The AUTOSAR standards have reached a new height, but will continue to grow and evolve to meet the today’s latest vehicle E&E requirements. For ECU software developers, embracing AUTOSAR in addition to enhancing their application features is often considered to be just one further burden that is too often undertaken only if mandated by the vehicle manufacturer. The proliferation of AUTOSAR architectures into powertrain areas has been particularly slower than in other domains (such as body and chassis applications) with one reason being the concern of handling the high-rate task scheduling demands in addition to all the new AUTOSAR features.

However, the mechanisms to handle such demands have already been considered and addressed by the AUTOSAR standards and many AUTOSAR applications are successfully running high-rate task schedules using the OS Cat1 interrupts. By analysing the processor load within such systems, Vector have observed that in many cases the high-rate task scheduling can still be achieved whilst also running the feature-rich AUTOSAR Basic Software. But inevitably the additional features that are provided within the AUTOSAR Basic Software and AUTOSAR OS will bring further demands in load onto the processor, and therefore new differentiators have emerged within the embedded software market. The scalability, configurability, efficiency and optimisation of each new AUTOSAR module has to be considered when sourcing AUTOSAR Basic Software as off-the-shelf commodity parts.

Adopting AUTOSAR continues to be a concern for many ECU suppliers, often heightened by the perception of the complexity it can bring. But the complexity is inherently present within the system already, driven by the end-customer’s feature requirements and the underlying E&E architecture necessary to support those required features. AUTOSAR only provides the mechanisms and structures necessary to manage the already existing complexity; therefore an AUTOSAR system should not be viewed as a problem – but rather, that it provides the solution.

Image rights:
Vector Informatik GmbH / Vector GB Ltd.

Literature:

Links:
Website Vector: www.vector.com
Vector AUTOSAR Solution: www.vector.com/autosar/