atollic

RTOS overview

Basic concepts and benefits

Fenway Embedded Systems

Real-time operating systems

Embedded applications typically have two design concepts:

- 'main' Loop (Infinite)
  - Each "task" called from main loop sequentially
  - Interrupts perform time-critical jobs
  - Stack usage un-predictable
  - User manages task interactions

- Using a Real-Time Kernel
  - Allows application to be separated into independent parallel tasks
  - Message passing eliminates critical memory buffers
  - Each task has its own stack area
  - Interrupt communication with event flags and message

Real-time operating systems

- A **real-time operating system** (**RTOS**) is an operating system (OS) intended to serve real-time application requests.
- A key characteristic of a RTOS is the level of its consistency concerning the amount of time it takes to accept and complete an application's task; the variability is *jitter*.[1] A *hard* real-time operating system has less jitter than a *soft* real-time operating system. The chief design goal is not high throughput, but rather a guarantee of a soft or hard performance category. A RTOS that can usually or *generally* meet a *deadline* is a soft real-time OS, but if it can meet a deadline deterministically it is a hard real-time OS.[*citation needed*]
- A real-time OS has an advanced algorithm for scheduling. Scheduler flexibility enables a wider, computer-system orchestration of process priorities, but a real-time OS is more frequently dedicated to a narrow set of applications. Key factors in a real-time OS are minimal interrupt latency and minimal thread switching latency, but a real-time OS is valued more for how quickly or how predictably it can respond than for the amount of work it can perform in a given period of time.[2]

Source : wikipedia

Real-time operating systems/2

The RTOS determines which applications should run in what order and how much time should be allowed for each application before giving processor access to another process:

- manages the sharing of internal memory among multiple tasks.
- handles input and output to and from attached hardware devices, such as serial ports, buses, and I/O device controllers.
- sends messages about the status of operation and any errors that may have occurred

Source : Quadros Systems

Why use an RTOS?

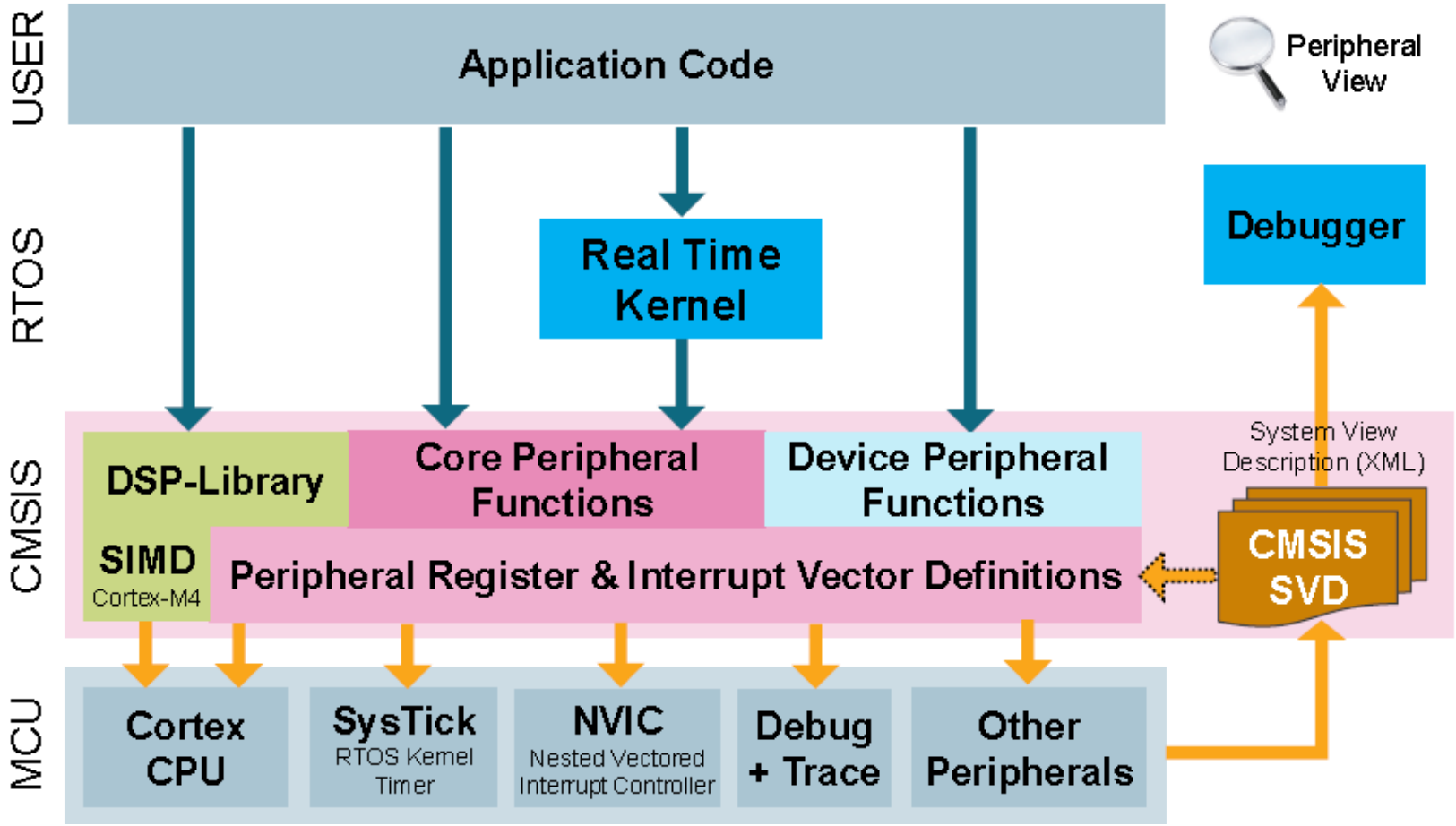
A well-designed RTOS provides a number of tangible benefits to the developer. It

- abstracts away the complexities of the processor,
- provides a solid infrastructure constructed of rules and policies that provide consistency and repeatability
- simplifies development and improves developer productivity by utilizing high level kernel objects to easily handle complex functions
- integrates and manages resources needed by communications stacks and middleware (TCP/IP, USB, SDIO, CAN, FAT and Flash file systems, etc.)
- optimizes use of system resources and improves product reliability, maintainability and quality

An RTOS can bring all those elements together into a platform that allows the application developer to begin development at a much higher point, enabling a shorter time-to-market with higher reliability and lower risk.

Source : Quadros Systems

CMSIS Version 2: Structure



Which RTOS?

HARD and SOFT real-time

- **Soft RTOS**

In applications involving soft real-time, timing constraints of those elements are looser than those of hard real-time to the extent that even a failure of a task to meet its time requirements still provides some value to the application. In essence, the soft real-time task does not offer a guarantee to meet its time constraint, but only that it will make a "best effort" attempt to do so.

  - in SOFT real-time systems, tasks are performed by the system as fast as possible, but the tasks don't have to finish by specific times

Source : Quadros Systems

Which RTOS? /2

HARD and SOFT real-time

- **Hard RTOS**

A task that has operational time constraints that must be met in order to avoid a catastrophic failure is called a hard real-time task. A system can have several such tasks and the key to their correct operation lies in scheduling them so that they meet their time constraints. That necessarily involves a priori setting of priorities to them and then analyzing each one with respect to the others to determine if a feasible schedule exists. A feasible schedule in a hard real-time system is one in which all tasks meet their known time constraints. In short, the basic property of the hard real-time elements of a system is that they are predictable.

  - in HARD real-time systems, tasks have to be performed not only correctly but on time

Source : Quadros Systems

Popular RTOS'es

This page has been left empty on purpose

Contact Us

- **Italy**

  Fenway Embedded Systems
  Via Don Giovanni Minzoni, 31
  20010 Arluno (MI) - Italy
  Tel. +39 02 97310120
  Email: sales@fenwayembedded.com
  Web: www.fenwayembedded.com

- **Headquarter**

  Atollic AB
  Science Park Jönköping
  Gjuterigatan 7

  SE-553 18 Jönköping – Sweden
  Email: sales@atollic.com
  Web: www.atollic.com

atollic ab
Science Park Jönköping
Gjuterigatan 7
SE-553 18 Jönköping
Sweden

atollic inc.
115 Route 46
Building F, Suite 1000
Parsippany
NJ, 070504
USA

www.atollic.com