Why Switch to an RTOS in Embedded Systems: Key Advantages for Microcontroller Projects

By Trevor Martin

Introduction

Many embedded system engineers working with small microcontrollers are used to writing firmware in a bare metal devices without using any formal framework. In this tech tip, we will look at some of the key reasons you should consider using a small-footprint Real Time Operating System (RTOS).

Don’t reinvent the wheel

An RTOS provides many common features that you would end up writing in a bare metal project such as delay loops, message passing buffer management etc. Once you are used to an RTOS you can use these features without the need to code them yourself.

Standardised API

The Arm Common Microcontroller Interface Standard (CMSIS) includes a specification for small footprint RTOS. The CMSIS-RTOSv2 specification defines a standardised API that is common to many free and commercial Real Time Operating Systems. This allows you to learn the API once and then use it many times. It also makes your code easier to port between different microcontrollers and allows you to select the best RTOS for your particular application.

Improved software architecture

One of the big advantages of using an RTOS is that it provides a framework that helps you develop code with a well-defined software architecture. Over time this is as important as the code functionality.

Structured code development
Similar to software architecture an RTOS allows you to develop code in a layered software model that aids improved project management and software code reuse.

Security
While you can use the TrustZone security peripheral with bare metal code the Arm Platform Security Architecture provides a set of security services designed to run in the secure world. Trusted Firmware for Cortex-M (TF-M) is intended to be accessed through a client running within an RTOS in the Non Secure World. This allows you to develop your application code as you would normally while all the sensitive data is held in the TrustZone secure world.

Better integration of 3rd party software
As your codebase gets more complex it will become necessary to integrate a range of third-party middleware libraries. This is easily done with RTOS threads that ensure each library is allocated its required resources and CPU run time. This builds a platform of services that can be accessed by your application code.

Multitasking
An RTOS lets you design multithreaded applications which use the RTOS scheduler to ensure each unit of code gets an appropriate amount of run time. This allows a developer to more easily design complex systems while the RTOS takes care of the execution details.

Debug support
While multithreaded code gives a programmer a lot of advantages and freedom it can be harder to debug and may also run into its own set of bugs. Using a debugger which has enhanced support for an RTOS based design can solve many of these issues. Debuggers can provide visibility of standard RTOS objects and also provide a visual trace of RTOS thread and interrupt execution.

Multicore support
There are an increasing number of microcontrollers which feature multiple Cortex-M cores. These are either symmetrical (ie two Cortex-M4) or asymmetrical (ie Cortex-M7 and a Cortex-M0). In response to this various RTOS vendors have started to include multicore support.

FuSa and functional isolation

Most of the popular RTOS' - such as Keil RTX and FreeRTOS - have a safety-validated version of the RTOS at a reassuringly expensive price. This allows you a big head start in developing a functional safety product. In the case of Keil RTX, it is also possible to develop a system with functional isolation. This allows you to run safety code and non-safety code on the same device minimising the certification requirements.

With Arm Cortex-M you can!

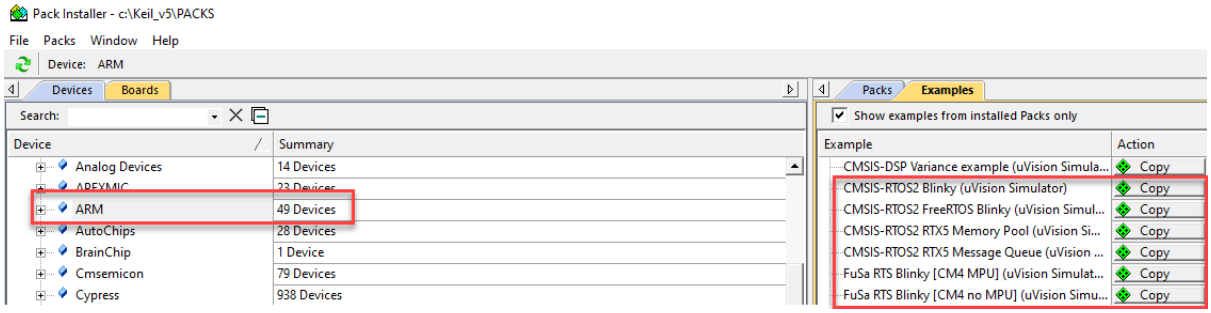
As the performance level of Cortex-M processors has continued to rise and the process feature size has shrunk the available number of CPU cycles and on-chip resources are no longer a limiting factor for most applications. The Cortex-M processor family also has a number of hardware and instruction set features which are intended to support the use of an RTOS.

How to start

The best way to start using an RTOS is to experiment with its features and then use it with a first small project. As your experience grows using an RTOS becomes an automatic choice even for the simplest of projects.

Pack examples

The device family packs contain some basic RTOS examples for Keil RTX there is also a Free RTOS pack that supports Cortex-M and Cortex-A. These are good starting points and give simple projects that demonstrate a minimal configuration. The RTX examples demonstrate the basic configuration and features of the RTX RTOS.



Training course

A full introductory tutorial and more advanced design techniques are presented in our one day RTOS training course. This is available as an in person course and also as a live online course.

The Designers Guide to Cortex-M

Much of the training course material is also available as a tutorial in my book “The Designers Guide to the Cortex-M processor family”

https://shop.elsevier.com/search?query=designers%20guide%20to%20the%20cortex-m%20processor%20family&type=Book

Further Information

For more information visit our website: www.hitex.co.uk or get in touch: info@hitex.co.uk. You can also connect with us: LinkedIn