



RealView Real-Time Library

Mark Onions

Product Manager Microcontroller Development Tools

RealView Real-Time Library.

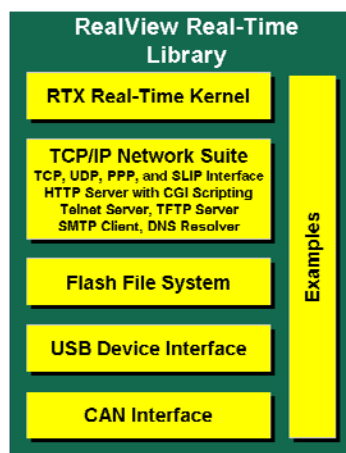
The embedded world is a complex world.

Today's embedded systems are expected to handle many tasks simultaneously whilst monitoring various interrupts with varying priority and be able to accept a wide variety of interface standards.

This presents software developers with real challenges; if they are to deliver the system on time, with the expected performance and the ability to expand with future market requirements and performance demands. These challenges increase when the system is also required to communicate efficiently with other systems, remote applications or via the internet.

One solution is the Keil RealView Real-Time Library (RL-ARM) offering a library of royalty-free, commonly used components to aid developers in developing modern embedded systems. RealView RTL-ARM includes:

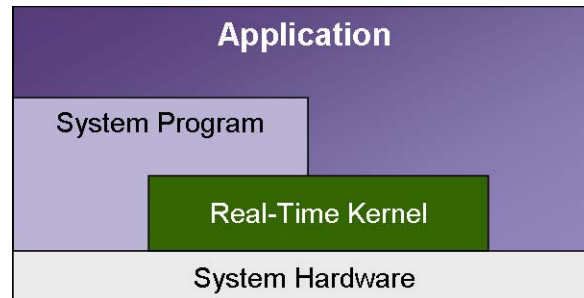
- The **RTX Real-Time Kernel**, a full-featured Real-Time Operating System that provides functions for: time management (for timeout or delay), semaphore management (for resource sharing), event management (for task synchronization), and mailbox management (for task inter-communication). The Kernel is easily retargeted using a single configuration file.
- The **TCP/IP Networking Suite**; a ground-up TCP/IP stack implemented specifically for embedded applications. It includes TCP and UDP sockets, PPP and SLIP interfaces, DNS, Telnet, TFTP, SMTP, and an Embedded Webserver with CGI and password protection. The stack works with Ethernet or Serial (Modem) interfaces. Preconfigured examples are provided for several standard evaluation boards.
- The **Flash File System** which allows you to save files in Flash or ROM (RAM support is currently in development). These are useful in systems that require large amounts of data storage or for systems with internet interfaces (HTTP or FTP).
- The **USB Device Interface** which implements standard USB devices such as HID or Mass Storage.
- The **CAN Interface** which provides a classic CAN driver that interfaces to the RTX Real-Time Kernel.



Real-Time Kernel

While it is possible to implement an embedded program without using a real-time kernel, a proven kernel like Keil RTX enables developers to save time, produce a reliable, expandable system and makes software development easier.

Using a real-time kernel allows developers to concentrate on application development rather than managing system resources, scheduling tasks and other general system 'house keeping', all of which are becoming increasingly complex when viewed in a system as a whole. It also creates an insulation layer between the application software and the hardware; allowing for future hardware upgrades and redesigns without requiring major software redesign.



Real-Time or not?

In this article we discuss the use of a real-time kernel (RTOS), unfortunately real-time is often confused with high speed. While real-time applications can certainly be high speed, real-time operation simply means that tasks or events should happen within a defined time period or in a predetermined order in relation to other events, Real-time systems can be relatively slow, a good example of this is a baggage belt at an airport, this is not fast, but it does have to ensure that the bags are sent to the correct location, this requires real-time control otherwise the bags will arrive at the wrong location.

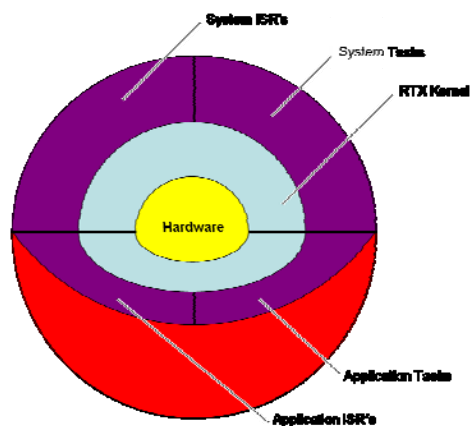
A real-time kernel should be viewed as an essential building block of a modern embedded system, allowing developers to create complex, flexible but easy to implement and maintain systems.

RTX Real-Time Kernel.

A good real-time kernel should offer the developer benefits in terms of performance and ease of use whilst remaining system friendly.

Predictable and scalable performance are a pre-requisite of a good real-time kernel; being deterministic in their behavior, an RTOS should ensure that tasks are completed within a known period and by offering low task latency this period should be kept to a minimum. A high number of interrupt levels (ideally more than 128), high number of active tasks and flexible scheduling schemes ensure that a good RTOS will cater for today's and tomorrow's system demands.

The royalty-free Keil RTX real-time kernel has been specifically designed to meet the demands of embedded systems and has been entirely developed and optimized for the ARM 32-bit architecture. It features deterministic behavior with context switching time $<5\mu\text{S}$, 255 interrupt levels a maximum of 256 active tasks and unlimited number of mailboxes, semaphores and user timers. RTX supports various scheduling schemes including cooperative, round robin and pre-emptive multitasking making it very flexible and an ideal choice for any embedded system.



RTX support is closely integrated in to the RealView Microcontroller Development Kit (MDK-ARM) which includes configuration wizards for all main parameters and full RTX debug awareness, RTX programs are written using standard C constructs and compiled with the RealView Compiler with in MDK-ARM. Additions to the C language allow you to easily declare task functions without the need for complex stack and variable frame configuration. All RTX libraries are automatically linked by the MDK-ARM IDE/debugger, μ Vision.

Although feature rich and powerful, RTX is very conservative with system resources, requiring only an on-chip timer, a minimum of 500 bytes of ROM and less than 5Kbytes of code RAM. Further more, full RTX source code is provided as part of Real-Time Library, allowing users to have full visibility of the code base used in their project, and ensuring they are able to fully maintain their application in the future.

Connectivity.

Almost all embedded applications are required to communicate with other systems or peripherals, most can be handled with straightforward serial interfaces such as USB and CAN. These are supported within Real-Time Library via a configurable device driver, providing users with an easily configurable tool to implement USB and CAN interfaces on a wide range of ARM based MCU devices.

Other some systems will require concurrent device and peripheral communication which cannot be handled via these types of serial interfaces, in these cases a well documented, mature, high performance interface with wide device support would be the ideal solution. The well known TCP/IP protocol meets this requirement perfectly and is essential if the system is to interface to a wider network or the internet.

However implementing a TCP/IP based system can be a daunting prospect for the embedded developer new to networking applications, where do they get a suitable TCP/IP stack, how do they implement it, how do they configure all of the complex peripherals, will it work with the other system components?

Fortunately life is not that complex when using the Real-Time Library. TcpNet is a ground-up design for embedded applications offering maximum performance, minimum memory requirement and ease of use. All common interfaces are supported for immediate use by the user, including TCP/IP and UDP socket interfaces as are various physical connection types including Ethernet, PPP (serial connection) and SLIP (dial-up).

As for the RTX real-time kernel, support for TcpNet is fully integrated in RealView MDK. Configuration wizards enable all networking parameters to be easily accessed and configured, allowing complex networks to be quickly implemented and tested. Full debug information can be output via a serial UART interface whilst the system is running; providing full visibility of all system networking activities giving the developer all the information required to verify correct operation of the networking interface with other system software components. TcpNet programs are written using standard C constructs and compiled using the RealView Compiler with in MDK-ARM (as for RTX).

Examples.

For the user, having the individual library components available to help them implement an advanced embedded system is a great step forward, however implementing them into a system and having them work together efficiently is another thing entirely.

This is actually a major strength of Real-Time Library, all components are designed to work together seamlessly, meaning that they can be used in a standalone manner or together as a complete system. The Real-Time Library makes it easy for developed to implement the components by including an exhaustive range of examples for the developer to evaluate and use. Examples cover real-time control, networking, web server and remote analysis applications, giving the embedded developer a good range of true to life example code which can be utilized in their applications if required.

As with all Keil products a high level of documentation is supplied with the product and further information is available at www.keil.com.

Conclusion.

The Keil Real-Time Library offers real advantages for the embedded developer in implementing modern systems while maintaining complete control of the system, complex networking and embedded connectivity are made easy and intuitive.

There are many other ways to accomplish the same result, however with Real-Time Library offering royalty-free implementation and a relatively low package cost of \$4195.00, there are few alternatives which give the developer such a comprehensive selection of components, all designed to work together and to be easily implemented into their end system.