

## Comparing different RTOS suppliers? Here are some important questions you should ask.

You've decided you need an RTOS for your next project.

How do you decide which RTOS to use?

Is it cost, features, footprint, gut feel, colleague recommendation or something else?

Many developers struggle when evaluating an RTOS decision. On the surface they can look very similar. But there are **fundamental differences** that you should know about.

- Do you know what questions to ask your potential RTOS supplier?
- In evaluating different RTOSes how will you weight the presence or lack of features?
- How do you put a value on development time or determine the cost of coding inefficiencies?
- How can you know how a particular RTOS will affect the performance of your system?

Here are some questions that may help you make a more informed decision.

- ***Does the RTOS give you a flexible set of scheduling policies?*** A well-designed embedded system may need a combination of two or more of these: Preemptive, Round robin, Cooperative, Time-sliced, Slicing with a variable other than time (e.g. angular rotation, flow, aperiodic or periodic ticks).
- All system objects (code entities, data passing, synchronizing, etc.) need to be established before the application can succeed. Most RTOSes can create static objects prior to runtime. ***Can your RTOS create dynamic objects at runtime? Just tasks or all objects?***
- ***Does your RTOS use the dynamic object's address as its identifier?*** This can lead to corruption of internal RTOS structures and possible failure.
- ***Are interrupts handled with a macro/function or do you have to write your own prologue (entry) and epilogue (exit)?*** The macro approach could save you a lot of development time – coding and testing.
- ***How does your RTOS synchronize with events? Do your event synchronization objects automatically clear with a task release or can events be missed? Can multiple tasks wait on a single event? Can a single task wait on multiple events?***
- Many RTOSes limit counting to ticks from a periodic interrupt to establish the system time. ***Any other form of counting is up to the user. Does your RTOS allow you to count both periodic and aperiodic ticks?***
- ***Can your RTOS count ticks other than time? Time may not be the only independent variable in the application.*** Counting angular position or displacement is useful for systems that have physical elements that are rotating at variable speeds such as engine control, ABS systems, turbine meters, etc. Ticks could measure flow rate through a positive displacement

meter. Don't force your system into a time-based structure if some other counting mechanism is a better fit.

- Some RTOSes use Timers; some use Alarms. The purpose is to initiate an action in the future. ***Are your Timers/Alarms globally available so that they can be used by multiple tasks or are they tied to a single task? Does the RTOS allow you to define future actions to be taken when any counter reaches a predefined value?***
- The best RTOS RAM management is usually done with pools/partitions in which each block is the same size. This prevents fragmentation and allows deterministic allocations.
  - ***Does your RTOS manage RAM with a heap that can create non-deterministic response and fragmentation?*** This can also add extra housekeeping overhead.
  - ***Can blocks from a memory pool/partition be used to create another partition dynamically?***
- Applications need to pass data from one code entity to another. There are many different names for this: Queues, Mailboxes/Messages, Pipes. Most data passing is either fixed-size or variable-size blocks.
  - ***Does your RTOS provide multiple data passing options?*** The more options, the better flexibility to meet your application requirements.
  - ***Does your RTOS allow data to be passed between tasks and ISRs (or just between tasks)?*** This improves system efficiency.
- ***How does your RTOS provide exclusive access to resources? Does it use binary semaphores or mutexes?*** Mutexes are generally better because they know the identity of the owning task.
- ***Does your RTOS have a mechanism to prevent priority inversion—when a low priority task has control of a resource that is required by a higher priority task?***
- Just because an RTOS is real-time doesn't mean it is fast. ***Ask your RTOS supplier about how the RTOS is coded? Is it designed for deterministic operation? Does it ensure low system overhead? Low latency? Responsive services?***
- Ask to see the RTOS API. Some RTOS products provide only a few primitive services while others incorporate a several hundred. The richer the API, the more likely it is that the RTOS can be easily configured to support your application.
- ***What does the RTOS supplier do to make it easy to begin development? What kind of getting started assistance is available? Does he supply sample projects tied to the development environment? Sample code? Examples of ISRs and drivers?***
- Memory footprint is an important issue. Some RTOSes are very small because they are very

basic. Others provide a rich environment that can be scaled to fit the requirements of the application. ***Can the RTOS be easily scaled down in size? Are there tools that automate this process?***

- Some RTOS products are fixed in size while others are scalable. Fine-grain scaling permits small adjustments to any included component such that the configuration will fit the application's needs
- Most RTOSes needs some form of configuration before they can be used. ***How easy is it to set up and configure the RTOS? Are there tools that automate this process?*** There are many configuration approaches. ***Which is best for you? Which saves you the most time and effort?***
  - ***Do you have to edit a source file manually?*** This can be easy to do but is prone to errors.
  - ***Does the RTOS use runtime code for creation of objects?*** This can involve a lot of tedious coding and is prone to errors and extra debug time.
  - ***Does the RTOS include a host-based (off-line) GUI configuration tool? Ask to evaluate the tool. What options does it give you? Does it allow you to configure both RTOS and application objects?***
- What about user documentation? ***How complete is it? Does it include clear descriptions of how the RTOS and its various objects work? Are there reference pages for each kernel service, including its description, calling and return parameters, any error conditions, and an example of each service?***
- ***Does the RTOS offer an integrated application design tool?*** Such a tool can save weeks of development. Its organized structure can help you better manage, understand and communicate your design to others.
- An RTOS can be compared to a professional carpenter's tools. An uninformed user could decide that a chisel could be used to tighten a screw not realizing that an electric screw gun with the proper tip was designed for just that purpose. ***What kind of training does the RTOS supplier make available?***
- Most commercial RTOS offer communications protocol stacks and middleware. This gives you the distinct advantage of a single point of integration and support for these products. But there are important questions to ask:
  - ***Some RTOS suppliers are resellers of third party software. What happens if there is a problem?*** Are they able to help you directly or will they just refer you to someone else.
  - ***How tightly is the software integrated? Does the RTOS supplier provide sample applications that integrate the pieces?***
  - The RTOS supplier may have the basic communications stacks and middleware you need

today, ***but what about the next project? Will they be able to support those requirements?*** Be careful not to strand yourself on a small island.

- ***Ask your RTOS supplier who handles support calls?*** The actual developers of the RTOS code are the best support team (they know the code know how to track down a problem).
- ***Do you have the confidence that today's RTOS will satisfy your future needs?*** A well-supported and maintained commercial RTOS can make future-proofing your application a reality.

Selecting the right RTOS for your project can make a big difference in your success. Minor differences between operating systems can add up to days, or even weeks of extra work for your development team. Be sure you ask the right questions before making a decision.

Quadros Systems, Inc.  
Direct: +1 832-351-2830  
Toll Free: 866 879-RTXC  
[www.quadros.com](http://www.quadros.com)  
[sales@quadros.com](mailto:sales@quadros.com)