

IntervalZero

IntervalZero ETS™ 14.1

Frequently Asked Questions about Real-Time

What is Real-Time?

Real-time computing describes systems that must respond to an event (or events) within a required upper bounded time frame. Typically, response times are in the millisecond or microsecond time frame.

What is the difference between “Hard” and “Soft” Real Time?

In a hard real-time application, the response to an event must occur before a certain deadline or the result is considered to be unacceptable.

In a soft real-time application, the response to an event should occur before a certain deadline. If it does not, the result may become increasingly more inaccurate, but may still hold some value.

What does determinism mean in a real-time environment?

Determinism is the ability to predict, with a degree of precision, when an event will happen. In a deterministic real-time environment, an event will occur within the required bounded time frame, and the performance of this event within the bounded timeframe is repeatable.

What is a real-time operating system, or RTOS?

A real-time operating system facilitates hard real-time and deterministic system development. The overall goal is to provide mechanisms for the developer that, if used correctly, will guarantee deadlines. Real-time goals are often confused with high throughput, which may not necessarily be a requirement for many real-time systems.

What is an Embedded Operating System?

An embedded system includes the computer hardware and software - including the operating system - that are built into a special-purpose device that is designed to perform a small number of dedicated tasks. Embedded systems are often expected to perform in real-time. They are optimized for performance by limiting functionality to only that needed by the device to achieve its intended design goals.

What are some examples of embedded systems?

Embedded systems range from the very simple to the very complex. Examples include management of complex flight simulators; control of manufacturing assembly lines; cell phones; test and measurement of motor vehicle response; systems which react to weather events or devices that change CDs in a jukebox.

What should I look for in an Embedded Operating System

The selection of an embedded operating system is dependent upon the task that needs to be performed. Some important things to consider are: What type of processor will be used? What are the memory needs? What kind of response time is required? How are processes scheduled? What type of external communication is required?

What is IntervalZero ETS™?

The IntervalZero Embedded ToolSuite (ETS™) provides the ability to create a complete real-time embedded runtime system for 32-bit x86-compatible microprocessors. The ETS SDK is a set of development tools that work in the familiar Microsoft Visual Studio environment. The ETS SDK uses the 32-bit Microsoft Visual C/C++ compiler and the IntervalZero LinkLoc linker to create an embedded real-time operating system (RTOS). The 32-bit Microsoft Visual C/C++ compiler and Microsoft linkers are used to develop an application to run on the RTOS. The combination of the embedded RTOS and application are referred to as an ETS Runtime. This ETS Runtime is then deployable as a complete standalone embedded system.

Frequently Asked Questions about ETS

How long has ETS been around?

ETS was originally developed by the former Phar Lap software company, founded in 1986. Phar Lap's products are now part of IntervalZero.

What is the current version of ETS?

The latest version of ETS is version 14.1, released in August 2009.

What types of industries or products typically use ETS?

ETS is used in a wide variety of products and vertical markets. Developers bringing to market an application that requires system control, determinism, or real-time performance with a small footprint can benefit from ETS.

ETS is used in some of the following markets:

- Industrial Automation
- Medical

- Military Aerospace

What are the benefits of ETS?

The real-time ETS kernel provides a small footprint, full-featured real-time operating system for running an embedded program. It provides a simple interface to develop programs that can be built into an embedded system. Features are included as individual components, enabling selection as needed. Development can be done with the familiar Microsoft Visual Studio environment.

How does the ETS SDK create a real-time image?

ETS includes a Visual System Builder (VSB) that provides a simple and reliable way to build an operating system that includes only what you need. The VSB interface lists available components and optional parameters. VSB combines selected information with the core ETS real-time kernel to create an embedded real-time operating system for your target.

How is ETS packaged?

The ETS SDK is a suite of tools that include a Visual System Builder (VSB), a real-time kernel, libraries to support operating system functionality, exported Win32 APIs for use within your ETS application and driver components to support specific hardware and operating system components.

Frequently Asked Questions about Platforms

Are there any hardware or platform requirements for ETS?

The ETS target Runtime runs on any Commercial Off The Shelf (COTS) x86 platform. ETS supports uniprocessor platforms. However, because all systems are not the same, developers need to evaluate the latencies of any platform that they choose to ensure that it can support their real-time or control needs.

Frequently Asked Questions about Development

What is needed for Application Development?

The ETS software development kit (SDK) provides a suite of tools that integrate seamlessly into the standard Microsoft Visual Studio IDE. The ETS Visual System Builder enables system developers to selectively choose components, effectively building the system from the bottom up. Because ETS was designed as a Win32 API compliant operating system, all the standard Windows conventions are maintained. This includes APIs, memory management, mutexes and semaphores that Windows developers are accustomed to using.

What is a split architecture?

The ETS split application architecture separates the system into separate ETS Kernel and application components. RTOS.EXE is the default name for the standalone ETS Kernel executable file. RTOS.EXE is built as the base operating system, with added functionality that allows it to load and start the execution of a separate, standard Win32-console style application.

How do I create an ETS RTOS?

After determining the hardware, storage and communication requirements for your Intel x86-based target, launch the Visual System Builder (VSB). Double-clicking the name of an item in the supported components list adds it to the list of selected components. Parameters can be set for selected components through individual property sheets. When configuration is complete, simply save the project to create a secondary boot monitor and base RTOS in your local directory. The RTOS is built using the Microsoft compiler and linked with LinkLoc, a linker-locator for embedded development. Application development is done using the familiar Microsoft Visual Studio environment.

Do I need special development and debugging tools to develop an ETS application?

No special development or debugging tools are needed to develop an ETS application. The ETS application code that will perform the embedded system's functionality can be developed using Microsoft Visual Studio. When the split architecture application is built, the Microsoft compiler and linker are used; LinkLoc is not required. References to ETS Kernel APIs are provided by the import library etsapi.lib, and are resolved from the ETS Kernel when the application is loaded.

How do I prepare the ETS system for loading onto the target?

Basic monitor settings in the Visual System Builder allow you to specify whether to build a monitor that can boot from disk, BIOS extension ROM or Boot Jump ROM. A boot disk monitor (diskmon) can be loaded onto a floppy disk, compact flash (CF), disk on chip (flash drive) or USB drive. The monitor is loaded onto the boot device either by using the ETS Boot Disk Wizard or by using DOS tools to place necessary files on the device.

How do I load the RTOS onto an ETS target?

When the Intel x86-based target is initially powered up, primary boot code, located in the BIOS, loads the ETS monitor. The ETS monitor continues the boot process by loading and starting the ETS Kernel on your target system. The kernel then loads and runs the application program.

Can I use Win32 API calls or are all ETS calls proprietary?

Your ETS application can use both ETS functions and a subset of Win32 API functions.

Does IntervalZero provide any ETS sample code?

The ETS toolkit includes source code for a number of sample applications. These samples can be compiled and run, and show important concepts and application interaction.

Are USB devices supported by ETS?

A USB component can be included in your ETS kernel, to provide support for USB devices.

In what ways is TCP/IP supported in an embedded ETS application?

Three flavors of TCP/IP support are available from the ETS toolkit.

- The initial TCP/IP stack includes configuration parameters in the kernel image but does not support multicast. It does support IPv4, SNMP v1, SMTP, HTTP, PPP, FTP, BOOTP, DHCP and DNS.
- The TCP/IP stack 2 component places IP parameter information in the RTOS initialization file, supports multicast and supports the IPv4 protocol.
- The TCP/IP stack 3 component places IP parameter information in the RTOS initialization file, supports multicast and supports the following protocols: IPv4, IPv6, FTP (IPv4), BOOTP (IPv4), SMTP (IPv4), DHCP (IPv4), DNS (IPv4) and SNMPv2c.

What Network Devices are Supported?

A file available at <http://www.intervalzero.com/ets.htm>, includes a complete list of supported network interface cards.

Frequently Asked Questions about Deployment

What is needed for application deployment?

Once a target image is created, and an embedded system is ready for deployment, each unit will require the purchase of an ETS Runtime license.

How can I help debug my customer's issues?

The ETS toolkit provides the option to enable debug message levels for the components that it ships. Debug parameters can be set to display error messages up through level 50.