

Taskrunner: A Method For Developing Real-time System Software

Mr. Louis M. Hebert
MIT Lincoln Laboratory

Abstract for
Sixth Annual Workshop on
High Performance Embedded Computing

A method is described for development of real-time system software that is independent of the operational, or target, hardware and its operating system. This method, called "TaskRunner" is based on experience with a variety of hardware and operating systems and has proven successful in debugging, testing, and deploying software modules used in the monitoring and control of complex instrumentation systems. Specific applications include airborne radar, multi-sensor aircraft instrumentation, and signal processing systems. The method supports development of multi-tasking, multiple-CPU systems.

The motivation for developing with this "middleware" technique is to allow code to be ported and maintained easily in an environment of rapid evolution of real-time hardware and operating systems. The simplicity of the programming interface (less than fifty methods) permits new programmers with no platform-specific experience to participate in development of complex systems with little additional training. Use of TaskRunner encourages modular, object-oriented code design.

The TaskRunner method consists of two main parts. First, a system-dependent library, written once for each target real-time system, implements a set of methods that conform to the TaskRunner specification and provide communication and scheduling services. The second is a "Task" library, which uses a corresponding set of methods to dispatch events and messages. Project development with the TaskRunner system consists of creating a set of polymorphic, system-independent Task libraries, which override the default methods of the basic Task and implement a functional module of the project. The advantage of this method is that the Task code will compile and run on any system that implements the TaskRunner library.

The TaskRunner library is a high-level interface, built around event-callback functions, rather than a set of function wrappers for common system routines. As such, the TaskRunner implementer can take advantage of the processing strengths specific to the target platform.

TaskRunner is not an exclusive method for a real-time project. A project may be made up of both TaskRunner and non-TaskRunner modules. Existing projects can be augmented with TaskRunner modules. It has proven to be fairly simple to retrofit TaskRunner to an existing project.

A TaskRunner library is implemented as part of a desktop emulator that allows for real-time, pseudo-real-time, and virtual-time debugging and testing of the component Task modules. The emulator application is available for the Windows, Solaris and Linux operating systems. Team development of software is facilitated with the use of the emulator when use of the target hardware for test is at a premium. The emulator is also useful for development of a system's user interface, and for user training. Project-specific tasks are loaded into the emulator through the use of a simple script which configures communication paths and other shared resources for related Tasks.

Taskrunner: A Method For Developing Real-time System Software

Mr. Louis M. Hebert
MIT Lincoln Laboratory

Abstract for
Sixth Annual Workshop on
High Performance Embedded Computing

A method is described for development of real-time system software that is independent of the operational, or target, hardware and its operating system. This method, called "TaskRunner" is based on experience with a variety of hardware and operating systems and has proven successful in debugging, testing, and deploying software modules used in the monitoring and control of complex instrumentation systems. Specific applications include airborne radar, multi-sensor aircraft instrumentation, and signal processing systems. The method supports development of multi-tasking, multiple-CPU systems.

The motivation for developing with this "middleware" technique is to allow code to be ported and maintained easily in an environment of rapid evolution of real-time hardware and operating systems. The simplicity of the programming interface (less than fifty methods) permits new programmers with no platform-specific experience to participate in development of complex systems with little additional training. Use of TaskRunner encourages modular, object-oriented code design.

The TaskRunner method consists of two main parts. First, a system-dependent library, written once for each target real-time system, implements a set of methods that conform to the TaskRunner specification and provide communication and scheduling services. The second is a "Task" library, which uses a corresponding set of methods to dispatch events and messages. Project development with the TaskRunner system consists of creating a set of polymorphic, system-independent Task libraries, which override the default methods of the basic Task and implement a functional module of the project. The advantage of this method is that the Task code will compile and run on any system that implements the TaskRunner library.

The TaskRunner library is a high-level interface, built around event-callback functions, rather than a set of function wrappers for common system routines. As such, the TaskRunner implementer can take advantage of the processing strengths specific to the target platform.

TaskRunner is not an exclusive method for a real-time project. A project may be made up of both TaskRunner and non-TaskRunner modules. Existing projects can be augmented with TaskRunner modules. It has proven to be fairly simple to retrofit TaskRunner to an existing project.

A TaskRunner library is implemented as part of a desktop emulator that allows for real-time, pseudo-real-time, and virtual-time debugging and testing of the component Task modules. The emulator application is available for the Windows, Solaris and Linux operating systems. Team development of software is facilitated with the use of the emulator when use of the target hardware for test is at a premium. The emulator is also useful for development of a system's user interface, and for user training. Project-specific tasks are loaded into the emulator through the use of a simple script which configures communication paths and other shared resources for related Tasks.

Taskrunner: A Method For Developing Real-time System Software

Mr. Louis M. Hebert
MIT Lincoln Laboratory

Abstract for
Sixth Annual Workshop on
High Performance Embedded Computing

A method is described for development of real-time system software that is independent of the operational, or target, hardware and its operating system. This method, called "TaskRunner" is based on experience with a variety of hardware and operating systems and has proven successful in debugging, testing, and deploying software modules used in the monitoring and control of complex instrumentation systems. Specific applications include airborne radar, multi-sensor aircraft instrumentation, and signal processing systems. The method supports development of multi-tasking, multiple-CPU systems.

The motivation for developing with this "middleware" technique is to allow code to be ported and maintained easily in an environment of rapid evolution of real-time hardware and operating systems. The simplicity of the programming interface (less than fifty methods) permits new programmers with no platform-specific experience to participate in development of complex systems with little additional training. Use of TaskRunner encourages modular, object-oriented code design.

The TaskRunner method consists of two main parts. First, a system-dependent library, written once for each target real-time system, implements a set of methods that conform to the TaskRunner specification and provide communication and scheduling services. The second is a "Task" library, which uses a corresponding set of methods to dispatch events and messages. Project development with the TaskRunner system consists of creating a set of polymorphic, system-independent Task libraries, which override the default methods of the basic Task and implement a functional module of the project. The advantage of this method is that the Task code will compile and run on any system that implements the TaskRunner library.

The TaskRunner library is a high-level interface, built around event-callback functions, rather than a set of function wrappers for common system routines. As such, the TaskRunner implementer can take advantage of the processing strengths specific to the target platform.

TaskRunner is not an exclusive method for a real-time project. A project may be made up of both TaskRunner and non-TaskRunner modules. Existing projects can be augmented with TaskRunner modules. It has proven to be fairly simple to retrofit TaskRunner to an existing project.

A TaskRunner library is implemented as part of a desktop emulator that allows for real-time, pseudo-real-time, and virtual-time debugging and testing of the component Task modules. The emulator application is available for the Windows, Solaris and Linux operating systems. Team development of software is facilitated with the use of the emulator when use of the target hardware for test is at a premium. The emulator is also useful for development of a system's user interface, and for user training. Project-specific tasks are loaded into the emulator through the use of a simple script which configures communication paths and other shared resources for related Tasks.