



# ZiLOG Real-Time Kernel Product Brief

## Introduction

ZiLOG's real-time preemptive multitasking kernel, RZK, is designed for time-critical embedded applications. It is currently available for ZiLOG's eZ80Acclaim!™ family of devices that includes the eZ80F91, eZ80F92, and eZ80F93 microcontrollers and the eZ80190 and eZ80L92 microprocessors. RZK is royalty-free, configurable, scalable, and modular in design; it provides a rich set of features via easy-to-use and well-documented APIs. Additionally, RZK features are highly optimized to the stringent memory and performance requirements of typical 8-bit embedded applications.

RZK is available in a standard release package that includes an object library.

## Architecture

The ZiLOG Real-Time Kernel architecture follows a layered approach. The bottommost layer is the RZK Core, which provides some basic kernel services. Above the core layer are the RZK objects, which are used by the application via an API interface.

RZK objects use the kernel services of the RZK Core for resource management. These kernel services provide a set of easy-to-use APIs as an interface to the end-user application to manage different activities, as shown in Figure 1. The Resource Queue Manager facilitates the threads as they wait for resources, and the Time Queue Manager provides facilities such as the Timed Wait and Sleep functions. The Scheduler schedules and dispatches threads based on a scheduling policy. The RZK Core also features hardware-dependent files that are specific to particular architectures.

The RZK Object Layer comprises a number of optional RZK objects, such as threads, semaphores, and memory management. This modular approach reduces the final footprint of your application, which can optionally include certain RZK features based on the application's requirements. RZK services can be accessed by an application via a set of easy-to-use APIs. The layered architecture of RZK makes it scalable, and new features can be easily plugged in.

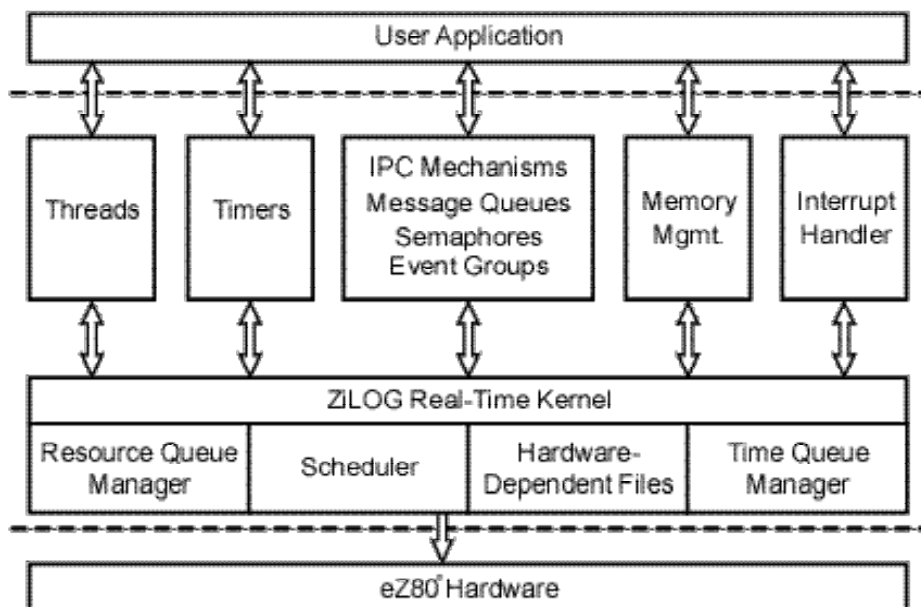


Figure 1. The RZK Architecture



## Features

ZiLOG offers features within its RZK release that are tailored to the requirements of typical 8-bit applications. The salient features of RZK are described below.

### Fast Context Switching

RZK can context-switch between threads in approximately 12 $\mu$ s (measured on an eZ80F91 MCU operating with 1 wait state at 50MHz). This interval includes the time required to identify the next ready-to-run thread as well as the time required to save and restore the context of the two threads, resulting in very low system overhead. Context-switching time exhibits negligible variations as the number of threads increases.

### Small Memory Footprint

RZK offers a very low memory footprint. The basic kernel and thread occupies only 8KB of ROM. Other objects can be optionally linked in. Additional objects are optimized for minimum memory requirements—the full version of RZK occupies about 18KB of ROM. The minimum RAM required is only about 1KB.

### Modular Design

The modular design of RZK enables the user to include only objects required by the application, thus saving memory space. RZK is released as a set of libraries, with each RZK object as a library file. Therefore, only the objects that are used in the application are included in the final downloadable image.

### Configurable

System parameters such as the number of objects (threads, semaphore, etc.) required can be specified at compile time. RZK allocates memory for the objects' control blocks based on the number of objects specified in the configuration file. Features such as Debug support and Priority Inheritances support can be enabled or disabled at compile time. Additionally, the duration of the RZK system timer tick is configurable. These compile-time configuration options make RZK fully configurable, thus

enabling the developer to fine-tune RZK to application requirements.

### Portable

Most of RZK is written in C. To boost RZK performance, a few of the critical routines are written in assembly. Only these files must be changed when RZK is be ported to a new platform. As a result, porting operations are minimized.

### Interrupt Handling

RZK manages all interrupt-related tasks. It provides APIs to install an interrupt handler for a particular peripheral device, and to conditionally enable and disable interrupts. The interrupt-handling mechanism allows the user to call RZK APIs from ISRs, and also supports nested interrupts. The minimum interrupt latency at 50MHz is 5 $\mu$ s.

### Threads with 32 Priority Levels

RZK provides preemptive as well as nonpreemptive threads. The threads can either preempt other threads based on priority, or share the CPU cooperatively among equal-priority threads using round-robin scheduling. Thread priority can be changed at run time.

### Supports Priority inheritance

RZK supports a priority inheritance protocol to solve a priority inversion problem common in binary semaphores. The low-priority blocked thread inherits the priority of the highest-priority thread blocked on that semaphore. RZK provides solutions for both unbounded and bounded priority inversion.

### Interprocess Communication

RZK offers various mechanisms for interthread communication and synchronization:

#### Semaphore

RZK provides both binary as well as counting semaphores. To emerge from deadlock scenarios, a priority inheritance protocol is used.

#### Message Queue

RZK provides message queues for asynchronous communication between threads. It supports buffered message copying. High-priority messages can



be placed at the front of the queue. Threads can wait on the queue based on their priority or FIFO.

### Event Groups

Event Groups are provided for control synchronization and do not contain any information. An event group can accommodate a maximum of 24 events. This aspect is very useful for synchronizing multiple threads based on specific events.

### Software Timer

RZK provides software timers to invoke periodic tasks at periodic intervals.

RZK maintains a system tick with the help of a hardware timer device that enables RZK threads to run the software timer, perform timed blocks on resources, and support clock functions.

### Memory Management

Fixed-size memory pools are provided for deterministic memory allocation, which is very useful for time-critical applications. RZK also supports variable-size memory allocation with features such as finite and infinite blocking.

### Demonstration Examples

The standard RZK package provides several sample applications that demonstrate various features of the RTOS for easy reference.

### Router Example

This application simulates a data router using relevant RZK objects to demonstrate the routing of packets and messages. The following RZK features are shown:

- Thread communication with message queues
- Thread synchronization with semaphores and event groups
- Use of timers
- Use of memory partitions

### Interrupt Example

This simple application demonstrates the use of an interrupt within the RZK environment to synchronize and generate the display of a message by an idle thread.

Other application examples demonstrate the use of the message queue, priority scheduling, and round-robin scheduling.

### Development Tools

RZK is supported by the ZiLOG Developer Studio Integrated Development Environment (ZDS II), which provides compiling, debugging, and project-building tools for the quick and efficient development of embedded applications. ZDS II is included in all eZ80 development kits.

### Packaging

The standard RZK package for eZ80 is supplied as a C object library module and application examples.

### Documentation

The following documents describe the features, functions, and usage of the ZiLOG Real-Time Kernel and are available in each RZK kit.

#### RZK Quick Start Guide

Enables the user to install the RZK software and quickly get up and running. It guides the user through a sample application.

#### RZK User Manual

Offers a detailed explanation of the different RZK configurations and sample applications. It also contains a section that analyzes RZK performance numbers and FAQs.

#### RZK Reference Manual

Provides a comprehensive explanation of the APIs and data structures provided by RZK.



## Ordering Information

RZK Package	Part Number	Description	Support
Standard Release	EZ800000100KRO	Object code/library package	Free support via <a href="http://www.zilog.com">www.zilog.com</a>

### Forthcoming Releases

ZiLOG plans to integrate the following features with RZK in upcoming releases.

#### RZK Device Driver Kit

A device driver framework which supports development of drivers for most of the peripherals on the eZ80 product line will be provided. Board support package consisting of drivers for most of the peripherals is also planned in forthcoming releases.

### Network Services

A full fledged TCP/IP stack is in development for RZK which will make offer basic as well as advanced networking features for low-cost embedded networking applications.

### Flash File System

A file system to manage the flash media.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, or to request copies of publications, contact:

### ZiLOG Worldwide Headquarters

532 Race Street  
San Jose, CA 95126  
USA  
Telephone: 408.558.8500  
Fax: 408.558.8300  
[www.ZiLOG.com](http://www.ZiLOG.com)

### Document Disclaimer

ZiLOG is a registered trademark of ZiLOG Inc. in the United States and in other countries. All other products and/or service names mentioned herein may be trademarks of the companies with which they are associated.

©2004 by ZiLOG, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZiLOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZiLOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. Devices sold by ZiLOG, Inc. are covered by warranty and limitation of liability provisions appearing in the ZiLOG, Inc. Terms and Conditions of Sale. ZiLOG, Inc. makes no warranty of merchantability or fitness for any purpose Except with the express written approval of ZiLOG, use of information, devices, or technology as critical components of life support systems is not authorized. No licenses are conveyed, implicitly or otherwise, by this document under any intellectual property rights.