zilog
An IXYS Company

eZ80Acclaim!® Family of Microcontrollers

Zilog Real-Time Kernel

Product Brief

PB0155010-0910

Introduction

Zilog Real-Time Kernel (RZK) is a real-time, pre-emptive, and multitasking kernel designed for time-critical embedded applications. It is currently available for Zilog's eZ80Acclaim!® family of microcontrollers. RZK 2.3.0 is now released as an integrated verion along with ZDS II—eZ80Acclaim! 5.0.0. RZK is installed automatically with the ZDS II—eZ80Acclaim! 5.0.0. installation.

Features

Key features of RZK include:

- Fast context switching
- Small memory footprint
- Modular design
- Configurable compile options
- Portable design
- Interrupt handling
- Threads with 32 priority levels
- Support for priority inheritance
- Interprocess communication using Semaphores, Message Queues, and Event Groups
- Software timer
- Memory management
- Device driver framework, which includes drivers for UART, EMAC, SPI, I^{2}C, WLAN, and RTC for eZ80Acclaim!
- Drivers for USB device
- Zilog File System

These features are customized to the requirements of typical 8-bit applications. The following sections describe these features in detail.

Fast Context Switching

RZK allows context switching between threads in approximately 12 µs (measured on an eZ80F91 MCU operating with one wait state at 50 MHz). This interval includes the time required to identify the next ready-to-run thread as well as to save and restore the context of the two threads. This results in a very low system overhead. Context switching time exhibits negligible variation as the number of threads increases.

Small Memory Footprint

RZK offers a very low memory footprint. The basic kernel and thread occupies only 9 KB of ROM. Other objects such as semaphores and threads are optionally linked to the kernel. Additional objects are optimized for minimum memory requirements—the full version of RZK occupies about 21 KB of ROM. The minimum RAM required is 1 KB.

Modular Design

The modular design of RZK allows minimal footprint increase when using the object library. The RZK binary image file is prepared in such a way that only objects required by the application are linked. Interdependency between modules is very minimal. Therefore, only the objects that are used in the application are included in the final downloadable binary image.

Configurable Compilation

RZK allows you to specify system parameters at compile time. For example, the number of objects, such as threads and semaphores required, are specified at compile time. RZK allocates memory for object control blocks based on the number of objects specified in the configuration file. Features such as *debug support* and *priority inheritance* support are enabled or disabled at compile time. In addition, the duration of the RZK system timer tick is configurable. These compile-time configuration options make RZK fully configurable, and enables you to fine tune RZK to application requirements.

Portable Design

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

Interrupt Handling

RZK manages all interrupt-related tasks. It provides application programming interface (APIs) to install an interrupt service routine (ISR) for a particular peripheral device, and to conditionally enable and disable interrupts. The interrupt-handling mechanism allows ISR execution in a separate thread to minimize worst-case system interrupt latency. Worst-case system interrupt latency for RZK is approximately 30 µs.

Threads with 32 Priority Levels

RZK provides preemptive and non-preemptive threads. These threads, either preempt other threads based on priority, or share the CPU among equal-priority threads using round-robin scheduling. Thread priority is changed at run-time.

Support for Priority Inheritance

RZK supports a priority inheritance protocol to solve a priority inversion problem common in binary semaphores. A low-priority thread that owns a particular semaphore inherits the priority of the high-priority thread, which is also trying to acquire the same semaphore. RZK also provides solutions for both unbounded and bounded priority inversion.

Interprocess Communication

RZK offers various mechanisms for communication and synchronization between threads. These are described in Semaphores, Message Queues, and Event Groups sections.

Semaphores

RZK provides support for both binary and counting semaphores. A priority inheritance protocol is used to emerge from deadlock scenarios.

Message Queues

RZK provides message queues for asynchronous communication between threads. It supports buffered message copying. High-priority messages are placed in front of the message queue. Threads wait on the message queue based on their priority or based on FIFO.

Event Groups

Event groups are provided for control synchronization and do not contain any information. An event group accommodates a maximum of 24 events. This mechanism is useful for synchronizing multiple threads based on specific events.

Software Timer

RZK provides software timers to invoke tasks at periodic intervals. RZK maintains a system timer 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 useful for time-critical applications. RZK also supports variable-size memory allocation with feature such as finite and infinite blocking.

RZK Device Driver Kit

A device driver framework is provided to support the development of drivers for most of the peripherals on the eZ80Acclaim!® product line.
This support includes a board support package

consisting of drivers for UART, EMAC, SPI, I^2C, WLAN, and RTC in all eZ80Acclaim! and eZ80L92 platforms, and also includes the USB device driver for eZ80Acclaim! MCU. It also provides drivers for different NOR Flash devices that provide simple-to-use APIs for interacting with devices equipped with Flash memory.

Zilog File System

RZK includes a file system to manage both RAM and Flash media on eZ80Acclaim!® platforms. This file system allows you to store files in RAM or Flash media and perform any file-related operations such as read, write, or append to the stored files. It also provides full-fledged directory operations, along with power-failure mode support.

Network Services

Using RZK, a full-fledged TCP/IP stack is developed, which offers basic as well as advanced networking features for low-cost embedded networking applications.

Architecture

Figure 1 displays the RZK architecture that follows a layered approach.

The lowest layer is the RZK Core, which provides basic kernel services as described below:

- The *Resource Queue Manager* keeps track of threads waiting for resources.
- 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 *Time Queue Manager* provides facilities such as the Timed Wait and Sleep functions.

Above the RZK Core layer are the RZK objects, which are used by the application using an API.

RZK objects use the kernel services of the RZK Core for resource management. These kernel services provide a set of easy-to-use APIs that allows your application to manage different activities.

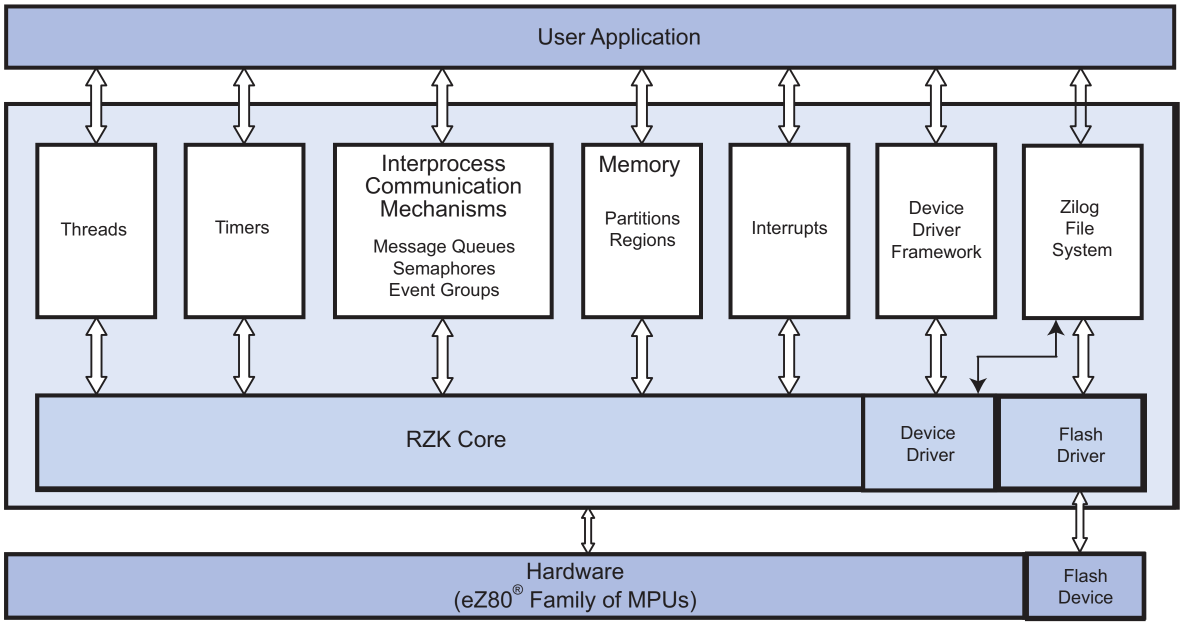

Figure 1. Zilog Real-Time Kernel Architecture

The RZK Object libraries comprises a number of optional RZK objects, such as threads, semaphores, and memory management. The design of individual modules inside the library is extremely modular. Unless an application specifically uses an RZK object, it is not included in the final binary image.

This modular approach reduces the final footprint of your application. An application accesses RZK services using a set of easy-to-use APIs. The layered architecture of RZK makes it scalable, and allows you to plug in new features.

RZK Sample Applications

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

Router Example

This application simulates a data router using relevant RZK objects on eZ80® to demonstrate the routing of packets and messages. The example provided below demonstrates the following RZK features:

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

Interrupt Example

This application demonstrates an interrupt in the RZK environment that is used to synchronize and display a message by an idle thread.

This package also contains examples that demonstrate the use of UART and RTC drivers. It also contains the interactive Zilog File System shell that implements file and directory operations that are similar to DOS commands such as `md`, `cd`, `dir`, `del`, `deldir`, and `deltree`.

Development Tools

RZK is supported by the Zilog Developer Studio Integrated Development Environment (ZDS II IDE), that provides compiling, debugging, project-building, and Flash loading tools for the quick and efficient development of embedded applications. ZDS II is included in all eZ80Acclaim!® development kits.

Packaging

RZK is packaged with the ZDS II—eZ80Acclaim! IDE. The standard RZK package for eZ80Accliam! microcontrollers is supplied as a C object library module with sample applications.

Documentation

The following documents describe the features, functions, and usage of RZK. These documents are available in each RZK kit:

- *Zilog Real-Time Kernel Quick Start Guide (QS0048)*—Enables you to install the RZK software. It guides you through a sample application.
- *Zilog Real-Time Kernel User Manual (UM0075)*—Provides a detailed explanation of the different RZK configurations and sample applications. It also contains a section that analyzes RZK performance numbers and FAQs.
- *Zilog Real-Time Kernel Reference Manual (RM0006)*—Provides a comprehensive explanation of the APIs and data structures provided by RZK.

Ordering Information

RZK Package	Part Number	Description	Support
Standard Release	NA	NA	Free support available on www.zilog.com
Source Release	eZ800000100KTS	ZDS II—eZ80Acclaim! 5.0.0	

Note: The RZK source and library are provided as integrated versions with ZDS II—eZ80Acclaim! 5.0.0.

Warning: DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

As used herein

Document Disclaimer