# Rodos (operating system)

From Wikipedia, the free encyclopedia

**Rodos** (**R**ealtime **O**nboard **D**ependable **O**perating **S**ystem) is a real-time operating system for embedded systems and was designed for application domains demanding high dependability.

**Rodos**

Rodos logo

| | |
|---|---|
| **Company / developer** | German Aerospace Center |
| **Programmed in** | C, C++ and Assembly language |
| **Source model** | Open source |
| **Supported platforms** | See #Supported Architectures |
| **License** | BSD license |
| **Official website** | dlr.de/rodos (http://dlr.de/rodos) |

## Contents



## History

Rodos was developed at the German Aerospace Center and has its roots in the operating system BOSS. It is used for the current micro satellite program of the German Aerospace Center. The system runs on the operational satellite TET-1 and will be used for the currently developed satellite BiROS.

Rodos is further enhanced and extended at the German Aerospace Center as well as the department for aerospace information technology at the University of Würzburg.

## Features

Rodos is realized as a Framework with a multilayer structure. The first layer is responsible for direct control of the embedded system hardware, on which the second layer containing the middleware runs. Task of the middleware is enabling a communication between different applications and components of the third, the top layer. Rodos was written object-oriented in C++, complemented by hardware-specific C and assembly code.

Rodos enables the user to write realtime applications for different architectures in an easy, efficient way. During the development special attention was paid to implement the various features of Rodos in a simple, nevertheless robust way. Unnecessary complexity was avoided to provide the user with a straightforward, clearly arranged system. Rodos supports typical features of realtime operatingsystems, like threads and semaphores.

Among other features Rodos offers:

- object-oriented C++ interfaces,
- ultra fast booting
- real time priority controlled preemptive multithreading,
- time management (as a central point),
- thread safe communication and synchronisation,
- event propagation

# Examples

## Hello World

The common Hello world example programm looks like this in Rodos.

```
#include "rodos.h"

class HelloWorld : public Thread {
  void run(){
    PRINTF("Hello World!\n");
  }
} helloworld;
```

As one can see, the implementation is straightforward. The class Thread is extended by a custom run() procedure, which writes Hello World to the standard output with PRINTF. All Rodos components needed for application development are accessible via the rodos.h header file.

## Threads

Rodos uses *fair priority controlled preemptive scheduling*. The thread with the highest priority is executed while running threads with a lower priority are paused (preemptive multitasking). If there are more than one threads with the same priority, each of them gets a fixed share of computing time and they are executed in turns.

### Example:

```
class HighPriorityThread: public Thread{
public:
  HighPriorityThread() : Thread("HiPriority", 25) {
  }
  void run() {
    while(1) {
      xprintf("*");
      suspendCallerUntil(NOW() + 1*SECONDS);
    }
  }
} highprio;

class LowPriorityThread: public Thread {
public:
```

```
  LowPriorityThread() : Thread("LowPriority", 10) {
  }

  void run() {
    while(1) {
         xprintf(".");
    }
  }
} lowprio;
```

The thread *LowPriorityThread* constantly writes the character "." and is interrupted every second by the thread *HighPriorityThread*, which writes the character "*".

## Topics

Rodos uses so-called *Topics* to enable communication between threads and over gateways between different systems. A *Topic* represents a message of a certain kind. A thread can publish *Topics* as well as subscribe to a *Topic* to receive all messages that belong to a type of message. The message system confirms to the publish–subscribe pattern.

Here is a simple example with one publisher and one subscriber, which both use the *Topic counter1* containing only one integer value.

### Example:

```
Topic<long>    counter1(-1, "counter1");

class MyPublisher : public Thread {
public:
        MyPublisher() : Thread("SenderSimple") { }

        void run () {
                long cnt = 0;
                TIME_LOOP(3*SECONDS, 3*SECONDS) {
                        PRINTF("Publish: %ld\n", ++cnt);
                        counter1.publish(cnt);
                }
        }
} publisher;

class MySubscriber : public SubscriberReceiver<long> {
public:
    MySubscriber() : SubscriberReceiver<long>(counter1) { }
    void put(long &data) {
        PRINTF("Received: %ld\n", data);
    }
}subscriber;
```

The *Publisher-Thread* post every three seconds an ascending counter value, while the *Subscriber-Thread* simply displays the received integer value.

# Supported Architectures

Supported processor architectures:

- Microcontrollers with ARM7 architecture
- Atmel AVR32
- STM32 32-bit ARM Cortex-M3
- Xilinx PowerPC PPC405
- Raspberry Pi

Furthermore Rodos can run as a guest on a different host operating system.

- Linux
- FreeRTOS
- RTEMS
- Windows
- TinyOS
- Posix

# External links

- German Aerospace Center – The RODOS-Framework (http://www.dlr.de/irs/en/desktopdefault.aspx/tabid-5976/9736_read-19576/)

Retrieved from "http://en.wikipedia.org/w/index.php?title=Rodos_(operating_system)&oldid=564989109"
Categories: Embedded operating systems | ARM operating systems

---