OCERA White Paper

OCERA White Paper
by A. Crespo, I. Ripoll

Published April 2003

Table of Contents

Index of Tables

Document Presentation

Table 1 Project Coordinator

Organisation	UPVLV
Responsible person	Alfons Crespo
Address	Camino Vera, 14 46022 Valencia, Spain
Phone:	+34 963877576
Fax:	+34 963877576
Email:	alfons@disca.upv.es

Table 2 Participant List

Role	Id.	Participant Name	Partcipant acronym
CO	1	Universidad Politecnica de Valencia	UPVLC
CR	2	Scuola Superiore Santa Anna	SSSA
CR	3	Czech Technical University in Prague	CTU
CR	4	CEA/DRT/LIST/DTSI	CEA
CR	5	Unicontrols	UC
CR	6	MNIS	MNIS
CR	7	Visual Tools S.A.	VT

Table 3 Document version

Release	Date	Reason of change
1_0	15/04/2003	First release

Chapter 1. Introduction

The objective of this White Paper is to provide a general scope of the OCERA components detailing the project goal, their use to produce different kinds of real embedded real-time systems and the use conditions or license.

1.1 Purpose

This document describes a general scope of the OCERA components detailing the project goal, provides guidance in understanding its us to produce different kinds of real embedded real-time systems and the license criteria.

1.2 Scope

This document offers a guide in understanding OCERA project and the different ways of building embedded real-time systems using the OCERA components mapping the standardized profiles of the IEEE POSIX standards. Detailed information about the architecture and concrete components of the OCERA project is outside of the scope of this document.

1.3 Intended Audience

This document is intended to be used by systems engineers, technical managers and procurement officers.

1.4 Document Overview

This document is organized in the following ways: section two provides an overview of the OCERA project goals and the kinds of embedded real-time systems that can be build. Section three looks at Profiles. Section four looks at licensing..

Chapter 2. OCERA Goals

The goal of the OCERA project is to provide support to real-time applications in embedded systems based on the Linux kernel.

This support will be provided by a set of components that will be:

flexible: new schedulers will support a wide variety of applications from hard to soft real-time;

configurable and scalable from a small to a fully featured system;

robust, providing fault-tolerant and high performance;

portable to several hw/sw configurations.

The results of the project will be:

a library of open source software components for the Linux kernel, that support real-time embedded systems at the design and at the implementation phase.

a contribution to the improvement of the Linux kernel by participating in the evolution of an open community of developers.

The OCERA consortium will join the real-time Linux community by following the standard development and design methodology currently used in open source projects. In this way the OCERA consortium and the Linux community will both benefit of mutual collaboration; also, the results of the OCERA project will be maintained and developed after the end of the project. **GPL => Quality**

In this project, we will to provide support for:

critical real-time applications that need a very low response time and can be considered as "trusted" (that is, we can guarantee their correctness);

less critical and more complex real-time applications, that may not be trusted (that is we are not sure about their correctness).

2.1 OCERA Components

An OCERA component is defined as follows:

"A piece of software that brings some new functionality or feature at different levels in some of the fields: Scheduling, Quality of Service, Fault-Tolerance and Communications"

and can be one of the following parts:

a modification of the Linux kernel or RTLinux executive, which will be released as a patch file against a specific kernel version or integrated into the final OCERA kernel;

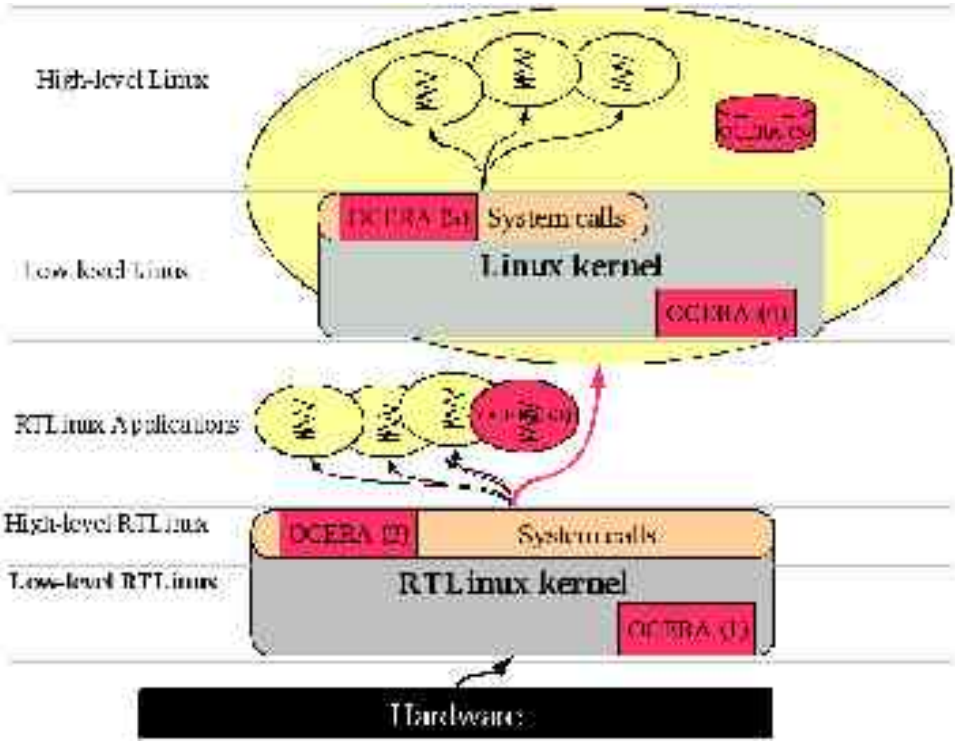
a module which can be loaded (with the insmod/modprobe commands) and provides new functionalities and may use some of the already installed services;

a library, dynamic or static, which can be linked with the user application;

a standalone thread or process (for example a debugging program).

2.2 OCERA Architecture and Component Classification

The OCERA architecture is based on the operating system Linux and the real-time executive RTLinux and allows to include OCERA component at different layers. Next figure shows the architecture and the component location.



The levels at the RTLinux are:

Low-level RTLinux: The components located at this level are highly related with the current RTLinux capabilities or internal algorithms, thus it requires to modify the current RTLinux source code in order to provide the new functionality or to improve an available one. This kind of component are distributed in a patch-form and hopefully incorporated in the main stream of the implied kernel source.

High-level RTLinux: Components at this level only need the current RTLinux API or an extended API offered by other kernel component in order to implement its new functionality. It does not require to modify the existing

kernel source code or any low-level kernel component.

RTLinux Application: These components use the kernel API to provide a new service. It does not require to modify the existing kernel source code or any kernel component. The main characteristic of these components are that they are implemented as an application-level processes/threads, offering some kind of service to other processes/threads (as a kind of a classical UNIX daemon).

In a similar way, the Linux layer is also split in two layers:

Low-level Linux : Like the low-level RTLinux executive components, these components modify the current kernel and has to be distributed as patch files.

High-level Linux : Components located inside the Linux kernel that use but do not modify the Linux kernel code. Device drivers are components of this category.

The third level (Linux applications) coincides with user space applications.

Chapter 3. Embedded Systems based on OCERA

This section describes the different OCERA Architecture profiles that OCERA components will support.

A profile can be seen as a collection of one ore more specifications or functionalities that can be used to define a certain application environment. IEEE Std 1003.13-1998 is a family of four related real-time profiles ranging in size from the very small through a full-featured platform. The smaller profiles specify just that subset of POSIX interfaces needed to "clothe" from widely-used small kernels to multi-computer distributed systems to be built, such as those used in factory automation.

OCERA architecture can be customised to build any of the profiles defined by IEEE. Additionally, OCERA components can offer a more powerful functionalities in the fields of scheduling, quality of service, fault tolerance and communications to improve the embedded system technology.

3.1 Building Hard Real-Time Systems

The use of OCERA to build hard real-time systems covers the two smallest profiles defined by the IEEE: *Minimal Real-time profile* and *Real-Time Controller profile*.

The Hard Real-time system build from OCERA architecture is based on the RTLinux layer of the global OCERA architecture as shown in figure 2.

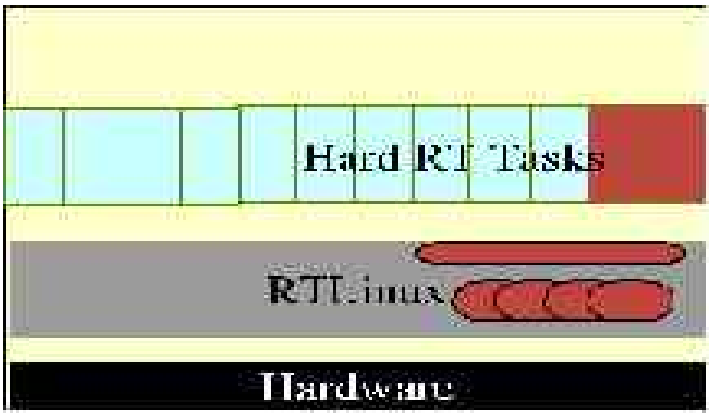


Figure 2. Hard Real-Time Architecture

The list of the optional features that can be included in this profile is:

Process Management Threads Threads communication & synchronisation Real-Time Signals Timers High Resolution Clock Semaphores Barriers Messages Scheduling Static Priority Scheduling Dynamic Priority Scheduling Application Scheduling Support Constant Bandwidth Server Feedback Scheduling	**Memory Management** Shared memory Dynamic Memory Management **Communications** Real-Time CAN Bus support CANOpen support **Fault Tolerance mechanisms** Fault Tolerant Controller Application fault tolerant monitoring Degraded task management **POSIX Tracing Facilities** **File System Support**

All this facilities can be selected and configured by means of a deployment tool.

This profile is a serious candidate for many kinds of real-time applications in the fields of Automotive, control systems, telecommunications, industrial automation, robotics, vehicle guidance, toys, railway and signal control, switching and telemeasuring, etc.

Multi-language Development Support for Ada and C.

3.2 Hard and Soft Real-time Systems

OCERA architecture is also designed to develop hybrid systems with hard and soft real-time activities. In this case, the user can allocate the critical task at the RTLinux level and the less critical tasks at the Linux level. The interface for both kinds of activities is a POSIX based interface.

This profile is intended for multi-purpose real-time applications as well as interactive users. It includes multiple processes, each of which may be multi-threaded. The IEEE profiles: *Dedicated Real-time Profile IEEE Std 1003.13 PSE53* and *Multi-Purpose Real-time Profile IEEE Std 1003.13 PSE54* can be easily achieved using this configuration.

At the RTLinux level the user can use the same components detailed in the previous

profile (hard real-time systems). At the Linux level, the provided functionalities are , additionally to the Linux functionalities, the OCERA components for this level summarised in the next features:

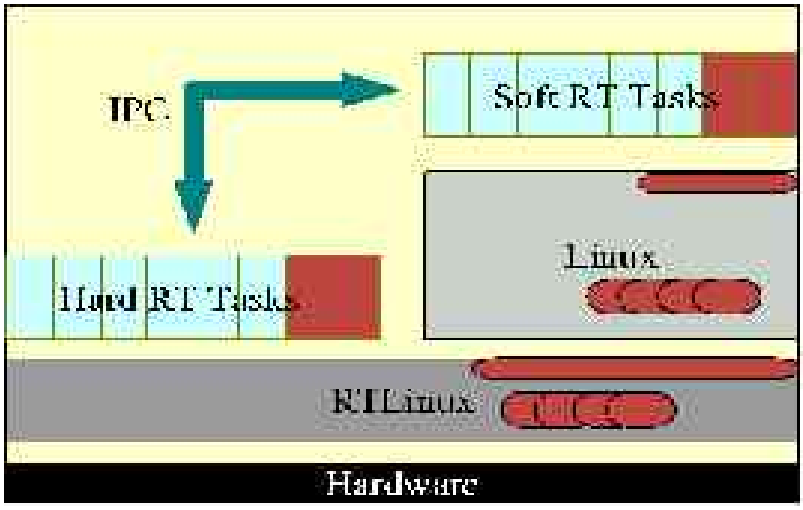


Figure 3. Hard and Soft Real-Time Architecture

Process management Threads Threads communication & synchronisation Scheduling Static Priority Scheduling Dynamic Priority Scheduling Constant Bandwidth Server Feedback Scheduling	Fault tolerance mechanisms Replica management Redundancy management Communications Real-Time Ethernet RTE support Real-Time CAN Bus support CANOpen support POSIX Tracing Facilities

All this facilities can be selected and configured by means of a deployment tool.

This profile can be used to applications with critical parts (implemented at the RTLinux level) and less critical parts (implemented at the Linux level) with the possibilities of all the Linux features.

The control of the non critical parts can be sintonised to guarantee a constant CPU bandwidth service. This is very useful in applications that require Quality of Service as multimedia application, robotics (environment maps generation), networking, etc.

3.3 Soft Real-time Systems

When no critical parts are defined in the application, the previous profile can be configured without the RTLinux level. In this case, the features are the described above.

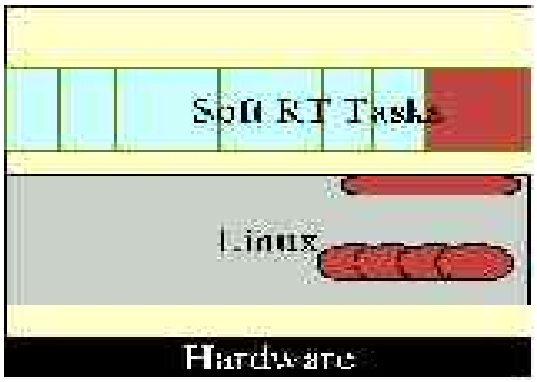


Figure 4. Soft Real-Time Architecture

Chapter 4. OCERA Tools

Additionally to the specific OCERA components, a set of tools provide support to the design, implementation and deployment phases. The tools are:

4.1 Design phase tools

ftdesign

The ftdesign is an off-line tool to help the user to specify the non-functional features of its application in a declarative way. This tool gathers information about the application task using fault-tolerant mechanisms and helps to build and instantiate the data structures needed for run-time management of fault-tolerance.

This design tool allows the user to express non-functional features. The first step will be the definition of a description language in order to specify explicitly timing characteristics and constraints of tasks, possible alternatives for tasks, actions to be done on temporal faults, and on errors. Some of the functionalities are:

Graph task definition

Support for imprecise computation

Critical task redundancy

Task modes

A result of this language definition task might be a UML profile for fault-tolerance. The acquisition tool itself will be developed using standard GUI programming tool.

PEP tool

PEP tool (Programming Environment based on Petri nets) is able to model concurrent systems and to verify them by partial model checking based on a compositional denotation Petri nets semantics. The language supported by the tools covers block structuring, parallel and sequential composition, synchronous and asynchronous communications and so on. Modeling allows to create either graphical version of Petri net model or structured program code of the model in B (PN)2 (Basic Petri Net Programming Notation) or SDL (Specification and Description Language).

PEP contains those verification components:

Deadlock-free tool - the tool checks whether or not a state can be reached in which no transition is enabled and displays corresponding transition sequence leading to such a state if one exists.

Reachability tool - the tool checks whether a (sub-) marking of a PN is reachable. The names of the places and the number of tokens on these places can be and the results of the previous test are displayed.

UPPAAL

UPPAAL allows modeling, simulation and verification of real-time systems. It is appropriate for systems that can be modeled as collection of non-deterministic processes with finite control structure and real-valued clocks, communicating through channels or shared variables. Typical application areas of the tool include real-time controllers and communication protocols in particular, those where timing aspects are critical.

4.2 Implementation phase tools

ftbuilder

The *ftbuilder* tool uses information gathered by the *ftdesign* tool and configures the OCERA fault tolerance support. It provides task information to the scheduler, instantiates FT mechanisms (AFT monitor and FT controllers), and adapts the code of the tasks to include the fault tolerance support. When task redundancy is specified, it produces the appropriated code for task duplication and check-pointing mechanisms.

4.3 Deployment phase tools

OF OCERA Framework

This tools is used to create a custom OCERA Embedded image, based on the requirements of embedded system hardware and software. OCERA component decomposition allows to achieve a highly granular level of customisation. The tool permits to customise the different OCERA layers selecting the required components to obtain an optimised image of the operating system and the application.

Through a set of menus, the user can specify the hardware, kernel components and application components to generate hard, soft and hard and soft embedded systems.

4.4 Validation and testing phase tools

Real-Time Graphic Analyser Tool

The RTGA is a graphical tracing tool to visualise an event log file. The log can be obtained from a traced system using the POSIX Tracing facility. The tool can analyse both the logical evolution of a traced system and the timing behaviour of the events.

This tool allows to visualise the task execution, kernel event occurrences, user defined events, etc.

Debugging tool

This is a low-level debugging tool (like gdb) that permits an external debugging of an OCERA embedded system through a serial port. This tool allows the step by step execution, access to the system variables, interrupt handler debug, etc.

Chapter 5. Legal issues

The OCERA project is a free software project. In what follows, it is assumed that the reader knows at least GPL and LGPL licenses[1]; the user that make use of the OCERA software is referred as *you*.

5.1 Legal status of Linux and RTLinux

Most, if not all, of the OCERA component depends in some ways on previous code. OCERA components can make **use** (link), **add** or **modify** already existing facilities of Linux kernel and RTLinux executive; also, some components can include code from other open projects in the form of linked library or verbatim copy. OCERA components must comply with the license usage terms and conditions of these programs and libraries.

In what follows, the licenses of Linux and RTLinux are briefly explained, and how these licenses impose some restrictions (or force some redistribution conditions) on the OCERA components.

5.1.1 Linux license

Linux kernel is distributed by Linus Torvalds under the GNU General Public License (GPL) version 2 license, with the explicit exception that:

> *This copyright does *not* user programs that use kernel services by normal system calls - this is merely considered normal use of the kernel, and does *not* fall under the heading of "derived work". Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the Linux kernel) is copyrighted by me and others who actually wrote it.*

The status of the Linux loadable modules is a GPL border case. The solution adopted by Linux developers is that any non-GPL loadable module loaded in the kernel will "*taint*" the whole kernel. A warning message is printed when a non-GPL module is loaded, Linux developers will not solve problems caused on systems that has been tainted, and also some kernel functions are not available to non-GPL modules. Core Linux developers (Alan, Linus, etc.) want to work only with GPL modules, and they consider as unpolite other licenses.

As summary:

User land applications can be distributed under any license, as Linux kernel is concerned.

Linux code (not modules) has to be GPL, or GPL compatible.

Linux loadable modules should be GPL, but can have any other license with severe restrictions.

1 http://www.gnu.org/licenses/licenses.html

5.1.2 RTLinux license

The original developers of RTLinux (Victor Yodaiken, Michael Barabanov) started-up a new company called FSM Labs.. This company distributes RTLinux code under two different license schemes: commercial and open/free.

1. Commercial: The name of the product sold by FSM Labs is RTLinux/Pro. FSM Labs offers a vide range of license types (developer seat, binary, source, runtime license, etc.). These licenses are similar to the 'standard' commercial licenses in the sense that the code is treated as a physical good the can not be copied, shared, given, etc. That is, to install on a machine a RTLinux/Pro system, the user must buy the appropriate license.
2. Open: The name of the product distributed by FSM Labs is RTLinux/Free, which is distributed under the 'Open RTLinux patent license Version 2' (*Open/RTLinux License* for short). This license allow you to use RTLinux without fee (for zero price) to develop:
   a) software licensed under the GPL; or
   b) software that executes within an unmodified Open RTLinux Execution Environment - whether that software is licensed under the GPL or not. In other words, it is possible to create proprietary products (and distribute them using a commercial license) using RTLinux/Free if and only if the original RTLinux code is unmodified.

The Open/RTLinux license terms has the following implications on the legal status of the OCERA, depending on the type of component:

If the OCERA component modifies the original RTLinux code then that component must be GPL.

If the OCERA component makes use of the original RTLinux API -do not modify the code but it is linked with RTLinux- then the component can be released under any license or licenses (open source or commercial). In this case, it is also possible to release this component under GPL.

Most of the the OCERA partners are think that GPL and LGPL is a good license to distribute components. You can expect that most of the OCERA components will be GPL.

5.2 RTLinux Patent

Software patents are not valid in Europe. Software patents is a relatively new legal framework developed in USA that do not has a counterpart in Europe. For this reason, U.S. Patents can not be enforced in European countries. Therefore, this section is only applicable to developments done by USA companies or users.

The mechanism used by RTLinux to take the full control of the hardware and which provides a virtual hardware to the general purpose Linux kernel is covered US Patent no 5,995,745(1999), by Victor Yodaiken.

As stated previously, FSM Labs distribute the RTLinux code (using the patented

code) under two license schemes: commercial and open/free. The commercial license grants to you the use of the patented technology as stated according to the license agreement terms. The legal situation of the open/free version of RTLinux was finally settled in October 2001. FSM Labs and the Free Software Foundation reached an agreement on the use of the patented technology releasing the 'Open RTLinux patent license Version 2'[2] which grants the right to use U.S. Patent No. 5,995,745 in GPL-covered free software without payment of a royalty.

5.3 Using OCERA system to develop GPL applications

Since Linux, RTLinux and OCERA components are GPL or compatible with GPL license, then **there is absolutely no problem to develop GPL applications**. As a brief summary:

If you do not use any OCERA component that modifies RTLinux original code (during the configuration of the OCERA system the user will be informed about the legal status and consequences of using each component) and your code do not modify RTLinux code, then the only requirement is to comply with the OCERA component license.

If at least one OCERA component is GPL then your code has to be GPL or *GPL compatible license*[3]..

If the OCERA components are LGPL the you can use any license for your code.

If you *modify* the RTLinux or use any OCERA component that modifies the RTLinux original code, then your has to be *GPL*.

Both RTLinux original code and OCERA components **are available on the web for free** -gratis, royalty free. You only has to comply with the GPL license as summarized above.

5. 4 Using OCERA system to develop non-GPL compatible programs

Since the OCERA system contain code from FSM Labs (licensed under the Open/ RTLinux license V2) and OCERA Consortium code, you have to obtain a non-GPL license from both parties. On one hand, FSM Labs will promptly sell to you a commercial license, but on the other hand, OCERA Consortium is more reluctant to do so. Please, get in contact with the OCERA project manager or with the author of the component.

There are another possibility to develop closed software. One of the two allowed uses of the Open/RTLinux license is the use of RTLinux with your program and distribute it under any license *if and only if* the original RTLinux code is not modified. The

2 GPL-compliant version of RTLinux Open Patent License in Works: http://www.gnu.org/philosophy/rtlinux-patent.html

3 GPL-Compatible, Free Software Licenses http://www.gnu.org/licenses/license-list.html#GPLCompatibleLicenses

OCERA consortium have achieved an agreement to include some of the OCERA components in the RTLinux code distributed by FSM Labs and covered by the Open/ RTLinux license (see section “RTLinux license”).

The OCERA installation utility will inform you about the conditions of each individual component, and the final license that you can use in your application.

Bibliography

[Baker91] T.P. Baker, 1991, The Journal of Real-Time Systems, 3, 67-100, *Stack-Based Scheduling of Realtime Processes*