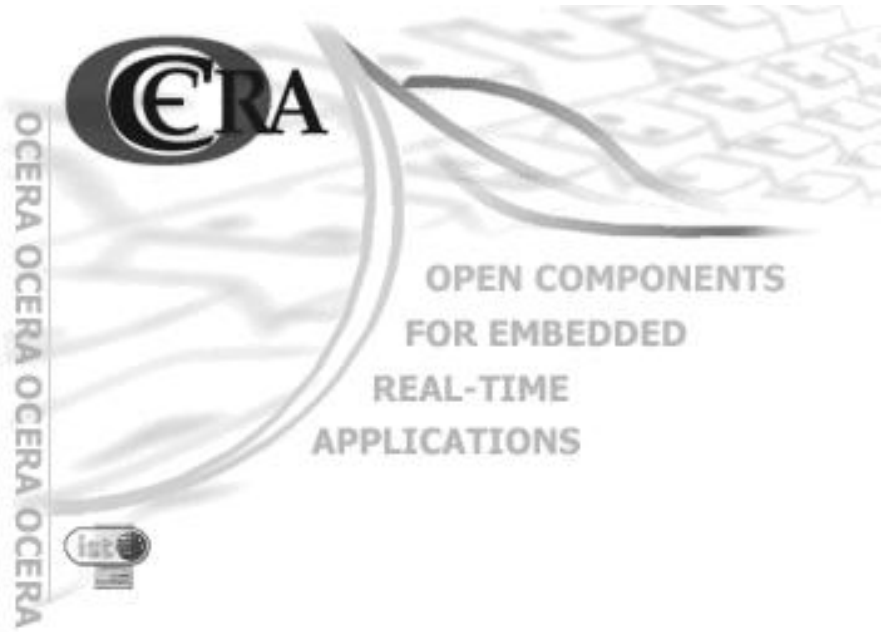
WP9 - Validation on platform



Deliverable D9rb.1 - Robotic Application Requirements and Platform Analysis

WP9 – Validation on platform : Deliverable D9rb.1 – Robotic application requirements and platform analysis

by F. Russotto and J. Brisset

Published February 2003

Table of Contents

1 General

The application we will develop is a servo-control application of an haptic device used into an Interactive and immersive system of virtual remote manipulation. This system allows manipulation by a human operator of virtual objects located into a virtual environment through an haptic device. The operator is immersed into the virtual environment using high screen projection and stereoscopic glasses and can interact with the virtual objects using the haptic device. The following figure shows an outline of the overall system :

Figure 1 : Interactive and immersive system of virtual remote manipulation

The haptic device used to manipulate virtual objects is a six-axis robot arm integrating six motors, six position sensors and a six-axis force sensor. It is controlled in position and force to give the operator force feedback computed from the interaction with the virtual environment.

The system involves three computers to operate : one computer for 3D virtual environment real-time rendering, one computer for virtual environment real-time dynamic simulation and one computer (also called Embedded Controller) for the servo-control robotic application of the haptic device.

The system is now operational and implements an Embedded Controller running the robotic application under VxWorks. We will first port the robotic application from the VxWorks OS to RTLinux-Ocera. Then, if the required performances of the robotic application can be fulfilled, we will port the virtual environment dynamic simulator to RTLinux-Ocera and try to run it on the Embedded Controller.

2 Requirements

The following chapter describes the requirements relative to the embedded robotic application that will run on the Embedded Controller. This includes : functional requirements, hardware requirements and software requirements.

2.1 Functional requirements

Four main functions can be identified from the functional analysis. The robotic application should provide :

- Cartesian control of the haptic device in position and force.
- Coupling interface to interact with the virtual environment.
- Virtual environment simulation (optional).
- User interface for Controller configuration and settings.

2.2 Hardware requirements

2.2.1 Architecture

The following figure shows an outline of the general hardware architecture (computer used for 3D rendering not shown) :

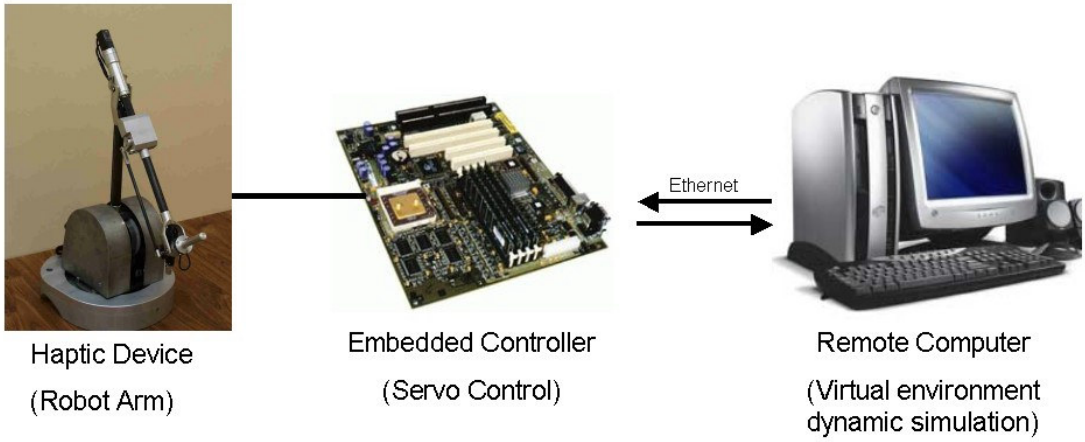


Figure 2 : General hardware architecture overview

The following paragraphs gives the detailed architecture of the main parts of the Embedded Controller.

2.2.1.1 Main board

The main board is a CPU NEXCOM 562 card. It is based on a x86 architecture (i440BX), supporting Intel Socket 370 CPU up to 1000 MHz. It integrates an onboard fast Ethernet 10/100 Mbps adapter and a PC104 (ISA) bus.

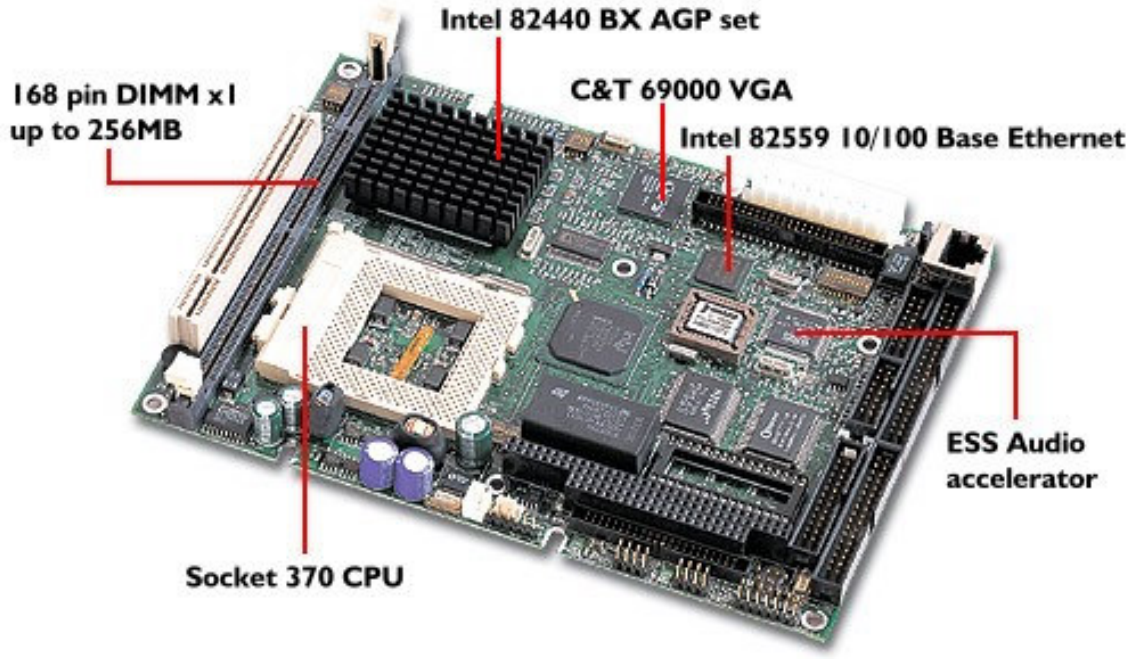
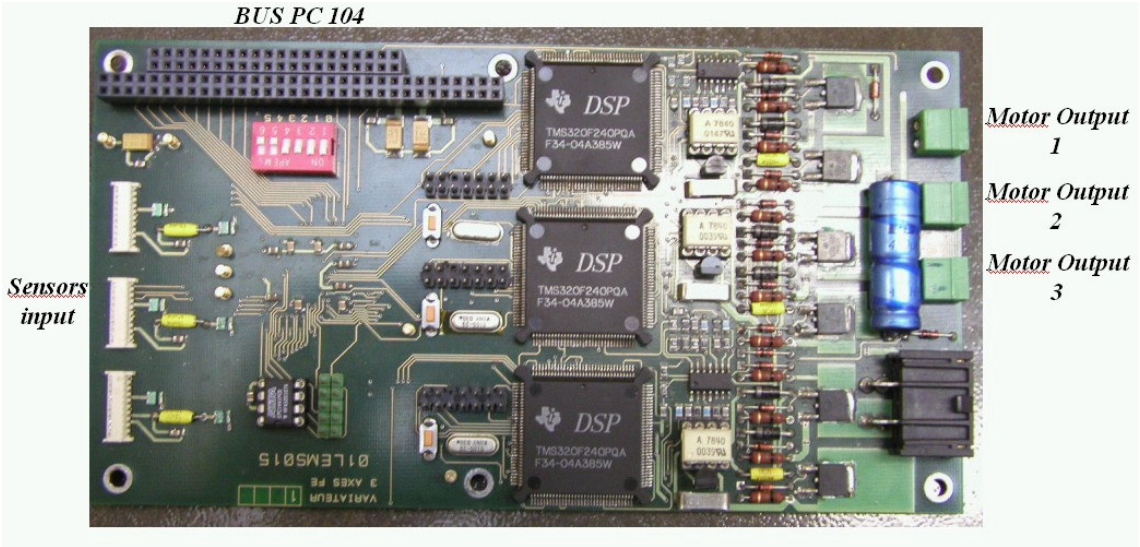


Figure 3 : Main board

The CPU is an Intel Pentium III running at 733 MHz.

2.2.1.2 Axis cards

Axis cards are intelligent Analog to Digital and Digital to Analog PC104 cards. These cards implement a TMS320F240 micro controller and a shared memory of 255 16 bits words.



Shared memory is accessible from both the DSP and the CPU through the PC104 bus. The following figure shows the general architecture of the card.

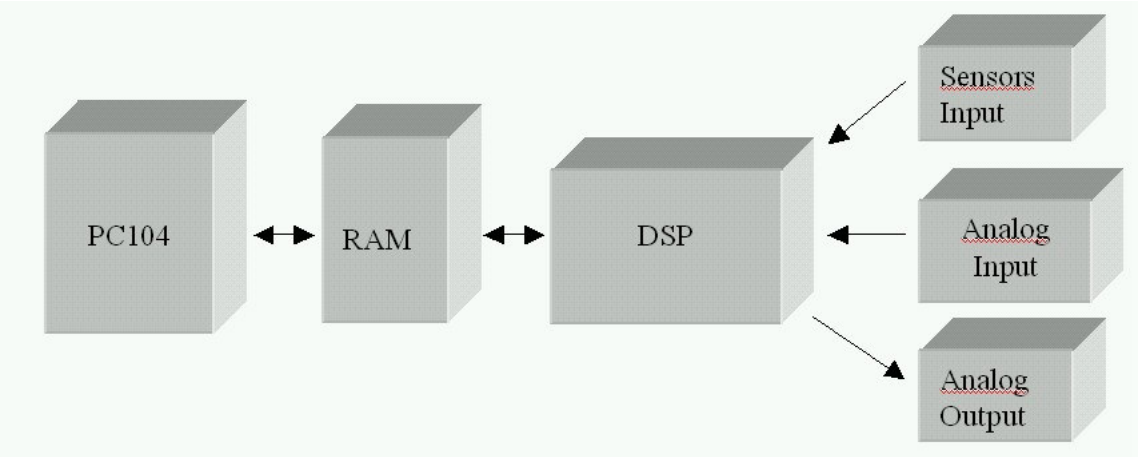


Figure 4 : Architecture of an Axis Card

The Axis cards are used to read angular position inputs of the haptic device and to write motor commands outputs.

2.2.1.3 DAC card

The DAC card is a standard PC104 card and is used to read from a 6D force sensor which is located on the terminal axis of the haptic device.

2.2.1.4 Logical I/O card

This card is a standard PC104 logical I/O card in charge of input / output logical (binary) data.

2.2.1.5 Power Supply

The Embedded Controller integrate a standard ATX power supply for the computer and a stabilized power supply MEANWELL SP500-48 for the Axis cards.

2.2.2 Requirements

The hardware requirements of the application are summarized hereafter :

- **Central Unit**

  The central unit should integrate a Socket 370 i786 architecture with a Pentium III processor running at 733 MHz or higher.

- **Input/output cards**

  The Controller should provide two Axis I/O cards (described above), an ADC Input card and a logical I/O card (as described above).

- **Communication**

  The Embedded Controller should provide a standard fast Ethernet 10/100 Mbps adapter.

- **Power supply**

  The Embedded Controller should provide a standard ATX power supply and a stabilized MEANWELL SP500-48 power supply.

2.3 Software requirements

2.3.1 Architecture

The following figure shows the general architecture of the application:

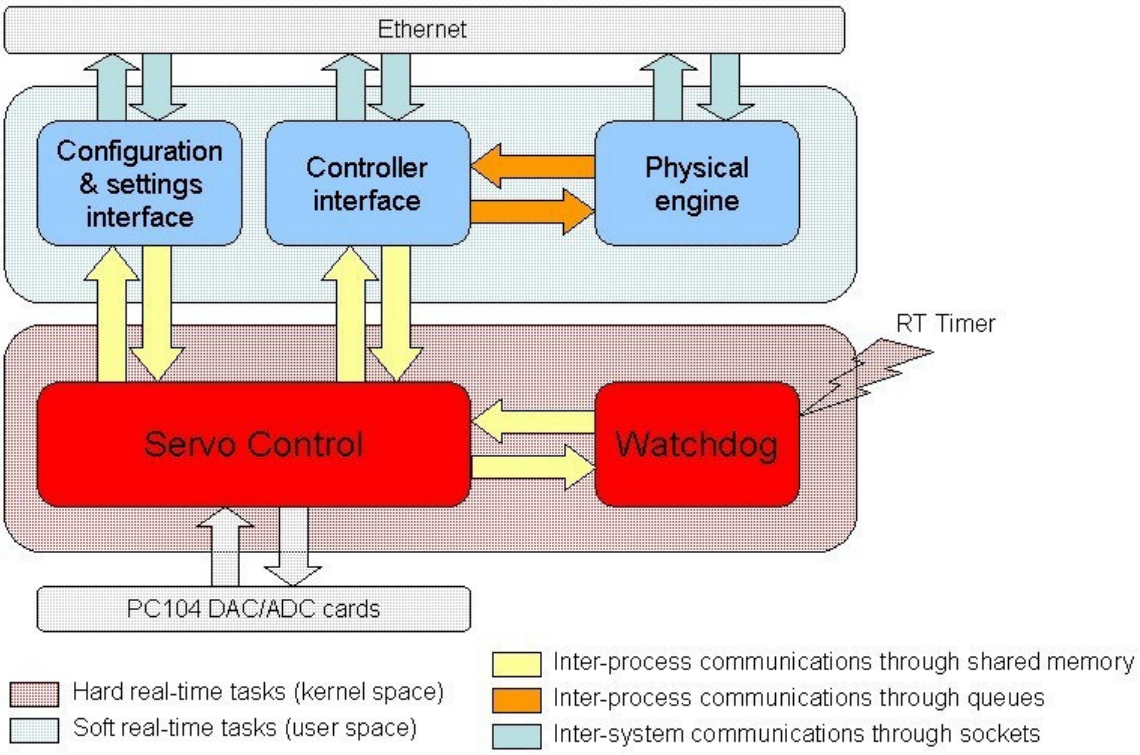


Figure 5 : Robotic application general architecture

The description of the job performed by each task is the following:

2.3.1.1 Servo Control

This is the main task of the overall application. This task performs servo-control of the haptic device in position and force. It is divided into six subtasks that are executed sequentially:

- Reading position and force from the haptic device sensors
- Receiving position and speed of the coupled virtual object from the Controller interface task
- Computing commands to be applied to the device motors
- Computing force to be applied to the coupled virtual object
- Writing commands to the motors
- Sending force to be applied to the coupled virtual object to the Controller interface task

This task is executed periodically (T = 1 ms) released by the Watchdog task using a synchronization semaphore.

2.3.1.2 Watchdog

This task is executed periodically (T = 1 ms) released by the system timer. The role of this task is to release the Servo Control task and to check that its execution time be in a 800 µs delay. If a deadline miss is detected by the Watchdog, it can stop and rerun the Servo Control task (signal not represented on figure).

2.3.1.3 Controller interface

This task is the coupling interface of the embedded controller. It opens a communication channel through a socket (protocol based on UDP/IP) and serves as an access point to remotely control the embedded controller. It communicates with the Servo Control task through shared memory.

In a first phase, this task will exchange information with the Physical engine task that will run on a remote computer connected through an Ethernet adapter. In a second phase, the Physical engine will run on the embedded controller and information will be exchanged through message queues (see figure).

2.3.1.4 Physical engine (optional)

This task performs a dynamic simulation of the virtual environment. One virtual object of the environment is dynamically coupled with the haptic device and the task exchange necessary coupling information through the Controller interface task.

In a first phase, this task will run on a remote computer (under Windows 2000). This phase will validate the application performances keeping the architecture that is now running under VxWorks. In a second phase, we will port this task to RTLinux-Ocera and will try to run it directly on the embedded computer as a soft real-time task (as represented on figure). We will then be able to evaluate application behavior (and so RTLinux-Ocera robustness) regarding the involved CPU load increase.

2.3.1.5 Configuration and settings interface

This task allows to remotely configure and set the controller parameters and characteristics. It also offer a mean to remotely trace internal variables changes and events of the controller. It communicates with other tasks through shared memory (only the communication with Servo Control is shown on figure).

2.3.2 Requirements

The software requirements of the application regarding the Real-Time Operating System are the following. They have been arbitrarily divided into the following four categories :

2.3.2.1 Scheduler

- **Low latency (context swap time < 15 µs)**

  A low latency is mandatory for the RTOS. Data acquisitions phase performed in the Servo Control task should start at a very precise time each period; a 15 µs delay is acceptable.

- **Fixed priority policy or better**

  The application now runs under VxWorks (which involves a fixed priority policy) with a good performance level. The scheduling policy could be different from the fixed priority one but the hard real-time tasks behavior should not be significantly impacted by the new policy; in short, no deadline miss is allowed to hard real-time tasks.

- **Optimized CPU usage for soft real-time tasks**

  During the second phase of the project, the Physical engine will be implemented on the embedded controller. Even if soft real-time, this task is very CPU hungry and requires from the RTOS as much CPU resource as possible.

- **Minimized priority inversion**

  The soft real-time Controller interface task will share many resources with other tasks (including hard real-time ones). This could result in priority inversion problems. The RTOS should offer a mechanism to minimize such issue.

2.3.2.2 Real-Time facilities

- **Tasks management**

  Common tasks management functions are required: task creation (periodic or one-shot), kill, lock and unlock.

- **High resolution timers**

  The acquisition phase requires a very precise clock top. A 1 µs resolution for the timer is required.

- **Semaphores / Barriers**

  Mutex semaphores and synchronization semaphores are used in the application to protect shared memory accesses and synchronize tasks. The RTOS should provide mechanisms satisfying these requirements.

- **Message queues**

  Message queues will be used to communicate between tasks.

- **Watchdogs**

  A common watchdog facility or equivalent mechanism should be provided.

2.3.2.3 Programming facilities

- **Dynamic memory allocation**

  Dynamic memory allocation is required for soft real-time tasks in such way real-time performances of the task is not degraded.

- **Memory protection**

  Because OS may run several tasks that could crash at any time, the robotic tasks (especially the hard real-time ones) memory should be protected from alteration by

these potential faulty tasks. A corrupted Servo-control task memory could involve physical injury and should be avoided.

- **Hard real-time tasks in user space**

  Debugging is very hard in kernel space. The RTOS should provide a mechanism to be able to run a hard real-time task into user space, at least for debugging phase. When performance loss involved is acceptable, running a hard real-time task into user space should even be preferable all the time.

2.3.2.4 Tools

- **Driver development guidelines and help tools**

  Robotics applications constantly require development of new drivers. Tools and/or development rules to make new drivers suitable to RT environment is required.

- **Debugging and tracing tools**

  Debugging is very hard in kernel space. The RTOS should provide mechanisms to help debug and trace tasks.

2.3.3 Performances

The required performances for the overall application are the following :

- **Servo control (hard real-time)**

  This task is released by the Watchdog task each millisecond. Its execution time should not exceed 600 µs in order for the computed motor commands to be valid. No missed deadline is allowed for this task

- **Watchdog (hard real-time)**

  This task is a periodic task launched by timer. It should start at the latest 15 µs after timer top. No missed deadline is allowed for this task.

- **Controller interface (soft real-time)**

  This task is continuously running, released by events coming from a socket. The task should be released at the latest 500 µs after a socket event. 10 % mean missed deadlines and 66 % peak missed deadlines are allowed for this task.

- **Configuration and settings interface (soft real-time)**

  This task is continuously running, released by events coming from a socket. The task should be released at the latest 100 ms after a socket event. A 1 s deadline miss is allowed.

- **Watchdog**

  This task is a periodic task launched by timer. It should start at the latest 15 µs after timer top. No missed deadline is allowed for this task.