
IST-2001-37652

Hard Real-time CORBA

Title

# RCT Design

Authors

Klas Nilsson (klas@cs.lth.se)

Reference IST37652/044 Deliverable D3.2
Date 2003-01-03
Release 0.95
Status Draft
Clearance Consortium

Partners

*Universidad Politécnica de Madrid*
*Lunds Tekniska Högskola*
*Technische Universität Wien*
*SCILabs Ingenieros*

# Summary Sheet

IST Project 2001-37652
HRTC
Hard Real-time CORBA

# Robot Control Testbed Design

**Abstract:**

Based on the requirement specification of the Robot Control Testbed for Hard Real-Time CORBA (HRT CORBA), such a testbed has been designed featuring

- Hard real-time distributed robot joint control using predictable communication by scheduled switched Ethernet
- Distributed CORBA encapsulated sensors for visual feedback, with soft real-time communication based on UDP / IP
- Simulation models enabling off-line evaluation of the impact on timing deficiencies on control performance

To be able to compare with CORBA and RT CORBA, standard transport over TCP / IP (or the ability to emulate the properties of TCP / IP) is also supported.

## Copyright



**HRTC Partners:**

Universidad Politécnica de Madrid
Lunds Tekniska Högskola
Technische Universität Wien
SCILabs Ingenieros.

# Release Sheet (1)

| | |
|---|---|
| Release: | **0.9 First Complete Draft** |
| Date: | 2003-01-03 |
| Scope | Initial version |
| Sheets | All |

# Table of Contents

# 1 Introduction

The envisioned CORBA features for hard real time should be possible to evaluate and illustrate in a robot control context. Furthermore, the effects of timing deficiencies in distributed computing and communication should also be possible to illustrate, thereby clarifying the motivation for HRT CORBA. The testbed will have to be based on the robot systems already available to the project, as further described in the RCT specification. The primary system to be used here is the one including an ABB IRB2000 manipulator, which has been completely reconfigured to permit control experiments. That is, the original control computers have been removed, hardware interfaces have been added, and our control computers have been connected. This forms a completely open controller, including servo control down to the torque control of AC motors, which requires several kHz sampling frequency for each of the six joints.

The system, depicted in Figure 1, has been working for some time. It was first based on M68k and DSP processors on a VME bus, and with synchronous RS422 communication with statically scheduled traffic to fulfil hard real-time

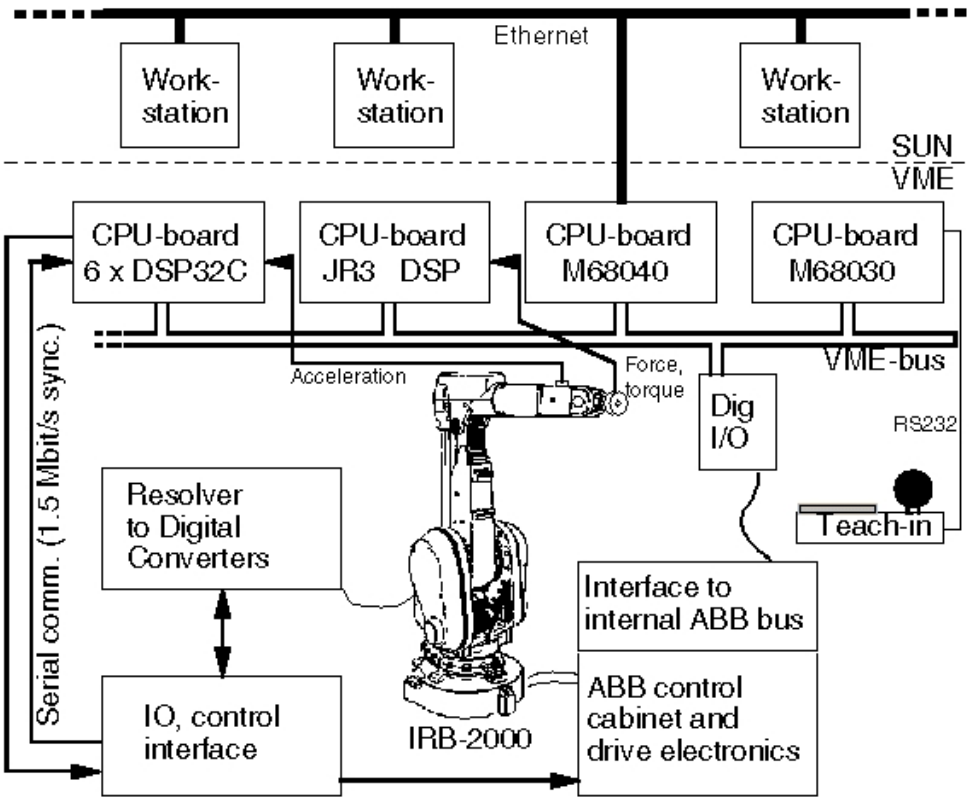


**Figure 1** The original reconfigured robot system.

requirements. Upgrading of processing and communication HW to more powerful and standard COTS was ongoing at the start of the HRTC effort, which added further considerations and requirements as presented here. In this section the basic design decisions for communication and computing is presented, while the more detailed design is presented in the following sections.

## 1.1 Communication

There are basically two attractive approaches for COTS-based real-time communication: Switched Ethernet (possibly with some twists for timing etc., called RTE hereafter) and TTP/C (or some alternative with the corresponding properties, called TTP hereafter). Preferably, to illustrate the HRTC pluggable transport, both RTE and TTP should be possible to use and replace with each other in a simple way. Of course, RTE cannot provide the same dependability and predictability as TTP, but from a CORBA point of view they should be manageable with different attributes reflecting the properties of the transport. It is also not clear whether the predictability of TTP is more valuable from a control point of view than the possibly higher bandwidth of RTE. Hence, experimenting with both techniques would be interesting.

It could be argued that a field-bus would be an alternative for real-time communication, since that is what they are designed for. The original RS422-based communication could be considered as a homemade field-bus, but aiming for commercially available HW there are several alternatives: Profibus, CAN, ControlNet, etc. However, all of these were ruled out for performance reasons. Most intricate was the ControlNet (refer to www.controlnet.org) that appeared to provide the necessary bandwidth and predictability, but a physical setup revealed that the static schedule could not be defined with cycles shorter than 1 ms. Since our minimum requirement was to be able to use at least the 4 kHz sampling frequency that the old system allowed, ControlNet was also rejected.

Initial experiments with 100Mbit/s RTE showed promising results, and the availability of inexpensive HW definitely makes this attractive. It turned out that 10 kHz sampling of all the robot joints was possible, and also very useful when experimenting with different control designs and real-time properties. Still with 10 kHz there was bandwidth left over, so transports with more redundancy and scheduling could be possible. Our initial aim for the RCT was to be able to plug in TTP as an alternative protocol to RTE. It turned out, however, that this was not possible. Our requirements implied the need for a TDMA-schedule of 100 µs, containing 4 slots as a minimum for the nodes listed, which is not possible with the current version of the C2-chip of the TTP/C HW. The length of one slot is calculated as a constant overhead per slot for

preprocessing and postcalculation of about 58 µs plus 0.32 µs per data-byte at a Bandwidth of 25 MBit/s (each slot introducing 4 bytes of protocol-overhead).

With 4 slots communicating 16 bytes in each slot a TDMA-schedule of about 250 µs duration is possible, which is the performance of the old system, giving no margin for experiments with fast sampling. In combination with the quite limited time for development of the testbed, it was decided only to use scheduled raw fast Ethernet for the RCT.

### 1.2 Embedded Computing

For general purpose computing and control, where availability of software is important but power consumption is not, we basically have only two HW alternatives: Intel Pentium and Motorola PowerPC (PPC), or processors compatible with these. For more supervisory (typically PC-based) control functions the Intel P4 provides excellent performance, and software availability is superior. For embedded hard real-time and predictable computing, the PPC is clearly preferable due to its shorter interrupt latencies. It also has a cleaner architecture and according to our experience it is also easier to work with when it comes to assembly-level debugging and device-driver development. Therefore, the PPC family is our choice for general-purpose processors that are to be interfaced with the controlled process.

For special purpose embedded computing and process interfacing we may look for more dedicated alternatives, for instance considering space, power, and IO ports. In our case, special processors may be appropriate for the distributed access to the internal sensors and actuators of the robot. Also for external (but so called intelligent) sensors, special processors may be appropriate, but that is basically part of the sensor system as such and perhaps software functions cannot be influenced anyway. On the other hand, an external sensor with built-in but open processing would be very interesting from a test-bed point of view. We found the Axis ETRAX to be an interesting special purpose processor, as further described below.

## 2 Static built-in control

The control built into the robot system is characterized by statically defined control and communication with minimum jitter and hard deadlines except for very few missed samples. The static definition means that the robot controller may need to be rebooted after reconfiguration, which is typically done in a setup mode.

## 2.1 Control and Real-Time Considerations

Hard deadlines and minimum jitter simply means that the control delay (from sampling to actuation) should (apart from being short for performance reasons) be almost constant, say within ±5% of the sampling period. For a sampling period of 250 µs this means max 25 µs variation. Based on classic control theory, such a fixed and known delay can be accounted for in the design, and experiences from industrial development of robot controllers suggest that this has been a good engineering principle.

However, if timing jitter could be accounted for using modern control theory, it would give more freedom in the design and cost optimisation of the system. In a system without jitter, jitter can be emulated by introducing varying delays in the actuation of control signals. In this way the RCT should be useful for testing various types of jitter. Note that while in process control the acceptable degree of jitter is mainly an issue for each individual control loop, in robot control it is the interaction between several servo controllers and the arm dynamics that is crucial. For instance, the jitter may introduce frequencies in the electro-mechanical motor system that in turn results in unacceptable noise and vibrations in the arm. This is hard to foresee since for instance the acoustic properties of the (hollow for cabling and stiffness reasons) arm are very complex. Hence, it should be possible to do experiments with different degrees of jitter as part of QoC measures.

If a few samples are lost, and that is detected (e.g. via time stamps of the sensor data) so the control can act accordingly, experience shows that this is not noticeable, and thereby it is acceptable. Depending on sampling rates and dynamics, the acceptable rate of lost samples (due to communication failure or missed deadlines) can be very different. Preferably only one out of several millions is lost, or less, but normally one lost sample out of 1000 does not cause any problem. The limits are, however, difficult to anticipate. Hence, the RCT design and implementation support permit loss of samples and its effects to be evaluated.

## 2.2 Distributed structure

The IRB2000 system should include the following distributed nodes:

- **Robot Joint Sensing (RJS) computer with sensor electronics.**
  Joint angles are measured via Resolver sensors, and the digital sensor value is obtained via RDC circuits (Resolver to Digital Converters, available from Analog Devices). There must be ports for handling six such circuits since there are six joints on the robot.§

§ Most robots use angular encoders, but a *resolver* provides absolute measurement within one motor revolution due to its construction with motor-like coils and working with induction. A

- **Robot Joint Control (RJC) computer with multiprocessor backplane.** The control computer performing servo control for all six joints will have to be part of a multiprocessor solution due to computing power needs for the full non-linear and multivariable control of the robot arm. To permit tight connection between the different CPUs, they interact via shared memory over a VME or PCI bus.
- **Robot Joint Actuation (RJA) computer with drive power electronics.** The actuator interface also requires some computing to receive actuation orders over the network and then control the motor drives accordingly. Motors are synchronous with three phases, requiring two current references (T-phase ref created in HW) to be output each 0.25ms. That is communication and timely output of 2·6·4000=48000 control signal values per second, each with 12 bits.
- **Process Value Sensing (PVS) node, optional, providing local HRT IO.** If there would be any remaining bandwidth on the internal communication network, it is very desirable to permit connection of customer sensors. On a predictable scheduled network, this means that unused time slots are made available for customer sensors.

In a more general mechatronic context, there could have been one sensor+actuator node per joint, and the joint control even more distributed. However, the original ABB system provides sensor connections to one location out on the robot (where our computer node replaces the original measurement HW), and actuator input to one location in the control cabinet (where our computer replaces the original 4xCPU +IO board). Therefore, the robot naturally forms one node and the control cabinet forms one node. Instead of the original computer board removed (for IO and programming reasons) from the cabinet, the motor control computer board is located in an external system, which then forms the third node.

momentary analog disturbance will therefore not result in a persistent error. This is also the reason why military equipment uses resolvers, or *synchros* that are basically the same but with built-in redundancy.

## 2.3 Computers and ORBs

The nodes described above are, as mentioned in the Introduction, connected via switched fast Ethernet. This means we have the structure depicted in Figure 2.

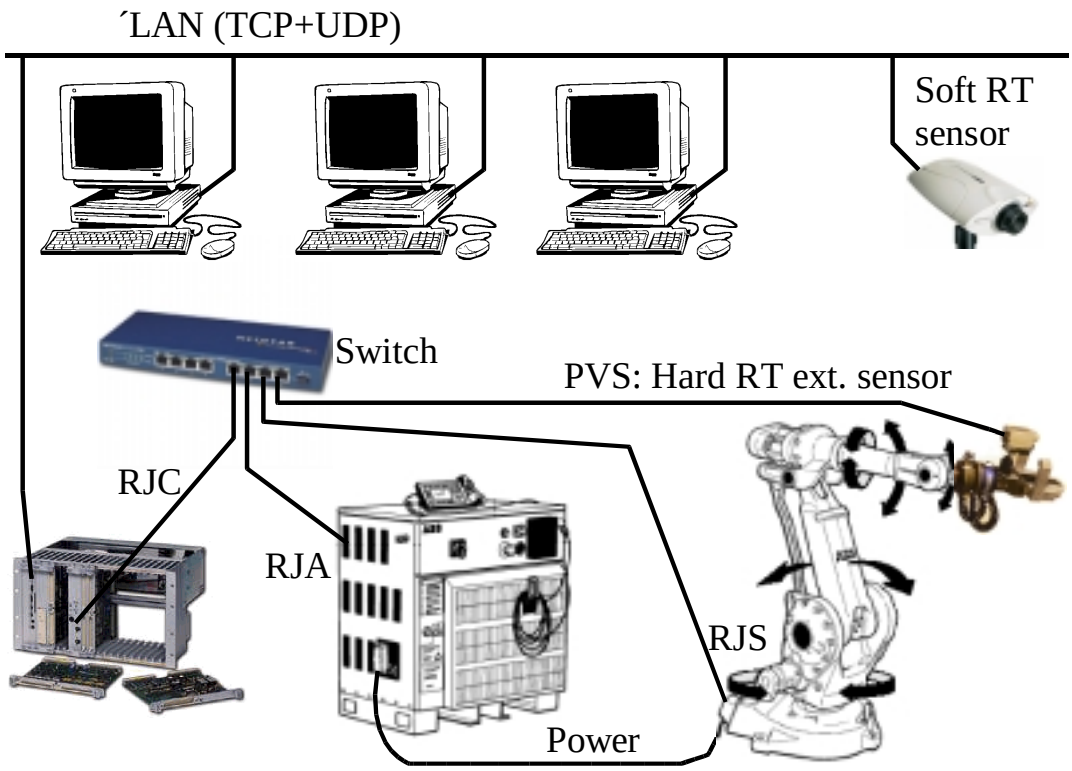


**Figure 2** Robot control nodes. The scheduled internal communication is connected to the switch. The high-level control is routed over a separate (non RT) network.

A minimum implementation is to have an ORB and CORBA objects encapsulating the RJC, and thereby also the total built-in control, for higher levels of control. Still, of course, the described communication and control issues have to be possible to investigate. It is, however, not clear what efforts are needed to fully HRT CORBA enable all of the involved nodes, but if possible the design is as described in the sequel.

## 2.4 Sensor and Actuator Devices

Both at the RJS and the RJA nodes, local IO signals have to be interfaced with the switched Ethernet. In the earlier version of the system with dedicated RS422 communication, specially built HW without processing power (but with programmable PGAs) was used since no affordable standard communication was available (ten years ago). Recently, the Axis ETRAX was found to be a suitable COTS alternative providing appropriate IO ports and a built-in fast Ethernet interface. It is a special low-cost type of CPU developed by Axis Communications (located next to the ULund HRTC partner), but both the

standard GNU C compiler and Linux have been ported to it. Also some of the application software (such as the communication parts) is open source. So, by using the ETRAX HW, Linux can be run both in the (processor added to the) mechanical robot arm, as well as in the interface to the drive power electronics. This also means a great deal of flexibility that should make it possible to add ORBs to the sensors and drives.

To implement HRT communication, there have been ongoing work (during fall 2002) to port the RTAI real-time extensions to the ETRAX+Linux platform. The aim for the RCT is to also include an ORB in these nodes.

### 2.5 Control Computers

For reasons mentioned above, the PPC was selected as the processor type for embedded HRT computing. There are then three issues that deserve special attention for the future success and availability (reproducibility at other sites) of the RCT:

- The SW platform needs to be suitable for real time, *and* based on an open and documented system with *standard* support for networking and the like. The real-time platform used for the robot control before the HRTC project was open but special to the ULund partner. That is, support and documentation for use at other sites was not sufficient. RTAI Linux now appears to be a suitable alternative. Thus, RTAI should be used as the control computer platform, possibly requiring porting if not available.
- The required backplane (shared memory) interface between the RJC and other control computers should be based on a standard bus. The current system is VME-based, while the most common bus today is the PCI bus. A special (physical) version of the PCI bus is the PMC (PCI Mezzanine Card) interface, which is also often used internally on both VME and PCI boards for interfacing different parts of the HW. The PCM is also expected to be commonly used in the future as a connector for additional computing hardware, e.g. for optional robot features. The IRB2400 system could hopefully also be interfaced to the same computer HW, since both PCI/PMC and VME/PMC adapters (or carrier boards) are commercially available. Hence, our preference is to use PMC computer boards.
- For control purposes, the PPC-G4 is very interesting due to its altivec unit, but no PMC PPC-G4 computer has (to our knowledge) been available. But around the start of the HRTC project, Motorola could deliver such a solution, the PrPMC800 board that also (optional) has built-in fast Ethernet.

The Motorola PrPMC800 is the most powerful and promising solution selected for RTC CORBA, but it requires suitable carrier boards and some changes to

Linux HW interfaces. As a backup alternative, current standard VME PPC boards can be used.

# 3 Application specific control based on external sensors

The built-in control treated above provides general-purpose motion control of the robot arm. That is, the control that the robot manufacturer is expected to develop. Basically, robots differ from other types of machines in that the specific application and control requirements are decided at the customer site. External sensors and system flexibility are therefore important issues. Most robots today are blind, but this is an obvious thing to improve upon, using CORBA techniques to improve on flexibility and engineering support.

## 3.1 Visual servoing

Visual feedback from our 3D and changing world to the robot motion control is quite demanding from a computing point of view, but very powerful if it can be tailored to user needs. Robot systems with feedback from video cameras have been around for some 20 years, but then as encapsulated products for 2D input to robot programs on the end-user level of the robot controller. Fully configurable 3D (or actually 6D including the identification of the orientation of objects) feedback to the servo control is, however, still in a research stage. Our experiences from this area points at the need for flexible distributed systems / objects with real-time capabilities. Visual servoing is well suited for being included in the testbed as a primary control application.

For 3D reconstruction of moving objects, stereo cameras are very useful. Two such video cameras, however, need to be synchronised in time to suit the computer vision algorithms, which adds timing requirements on the camera sensors. Video frames are captured with a 25 (or 30) Hz frequency, using ordinary cameras today. The sampling period of the control therefore is 40 ms. Also 80 ms can be useful if communication or computing resources are insufficient, and if that gives better total QoC. The visual servoing algorithm provides servo references to the built-in control. The specific algorithms are outside the scope of this design document; the sensors as distributed objects are the issue here.

## 3.2 Sensors, systems and ORBs

Ordinary composite video cameras with PC-based frame grabbers, as well as digital video cameras with FireWire interface, have been used in earlier experiments. However, none of these alternatives provides the synchronous frame grabbing needed for good stereo vision. A more promising approach would be to have a distributed system where the camera sensors and the control computers could be synchronised. That would not only serve as a

challenging testbed example, it would also be useful for the development of more powerful robot systems.

The motion controller is assumed to be HRT CORBA enabled according to Section 2, and computational objects and pluggable controllers (as mentioned in the Requirement Specification) are also clear from a design point of view. The remaining problem is the video sensor interface. As a backup solution in case of implementation problems, although deficient from a robotics point of view, is to connect video cameras to PCs running RTAI and an ORB according to the PCT. The aim is, however, the following design:

Cameras could be HRT CORBA enabled embedded devices, with an Ethernet interface that can be scheduled or using the UDP/IP protocol. The opportunity found is to use AXIS web-cameras that have the same type of processor and network interface as the sensor and actuator nodes of the built-in control. The same RTAI-based platform could therefore be used also for the external sensing, with the difference the timing requirements are less severe but having the device to concurrently also run the Axis camera software. Reconfiguration of the camera (new not yet public model) to support frame grabbing triggered from the network interface has been investigated, and in combination with predictable RT communication, the desired 3D visual servoing should be possible to accomplish based on interacting HRT CORBA objects.

# 4 Virtual testbed

A well-designed RCT will contain distributed objects that can be used also for non-real-time purposes, as part of tools for analysis or as parts of a testbed simulator.

## 4.1 Simulation Tools

TrueTime is a Simulink toolbox developed at ULund that makes it possible to simulate the timing behaviour of distributed systems consisting of nodes communicating over networks. The nodes are modeled as real-time kernel blocks supporting preemptive priority- or deadline-based scheduling of user tasks. The user tasks could, e.g., implement sensor and actuator interfaces and control computations. The user tasks could be interfaces to ordinary Simulink blocks modeling the process to be controlled. Different link-layer network protocols are supported, e.g., Ethernet (CSMA/CD) and TDMA. Within HRTC we are now extending the network model to also support, e.g., TCP/IP. Using TrueTime it is straightforward to evaluate how input-output latencies originating from, e.g., network delays effect control performance. It will also be possible to compare the performance of CORBA on top of TCP/IP versus the

performance of HRT CORBA on top of both RTE and TTP. A more detailed description of TrueTime and the simulaton-based virtual testbed is available in deliverable D1.1.

### 4.2 CORBA-enabled Virtual Robots

To make it possible to visualize and illustrate the testbed experiments, one (or several) portable computer(s), a corresponding virtual testbed will be developed. Since true real-time performance cannot be assumed to work on such computers, time driven software may need to be simulated in an event driven manner. The CORBA interfaces should of course be the same, and the dynamic behaviour should be close to that of the physical platform. Visualization will either be based to dedicated OpenGL-based graphical models, and / or implemented in Java3D in the case that the virtual testbed is Java-based. Initial test with both principles have been carried out, but decision about the best technique to use is postponed until implementation phase.

## 5 References

[CORB] Common Object Request Broker Architecture (CORBA/IIOP) Specification 3.01, Object Management Group, Needham, MA, U.S.A., 2002, http://www.omg.org

[RTCORB] OMG: Real-Time CORBA 1.0 Specification, http://www.omg.org

[InCORB] Mowbray, T. J. and W. A. Ruh: "*Inside CORBA*", Addison Wesley, 1997.

[Nil99] K.Nilsson and R.Johansson: *"Integrated architecture for industrial robot programming and control"*, Robotics and Autonomous Systems, 29, 205-226, 1999.

[PCI] PCI Special Interest Group: "PCI Local Bus Specification", 1998, http: / / www.pcisig.com

[PrPMC] VITA Standards Organization (VSO): "Processor PMC Standard For Processor PCI Mezzanine Cards", 1999, http: / / www.vita.com

[ETRAX] Axis Developer Board LX, Axis Communications, 2002, http: / / developer.axis.com / products / devboard / index.html