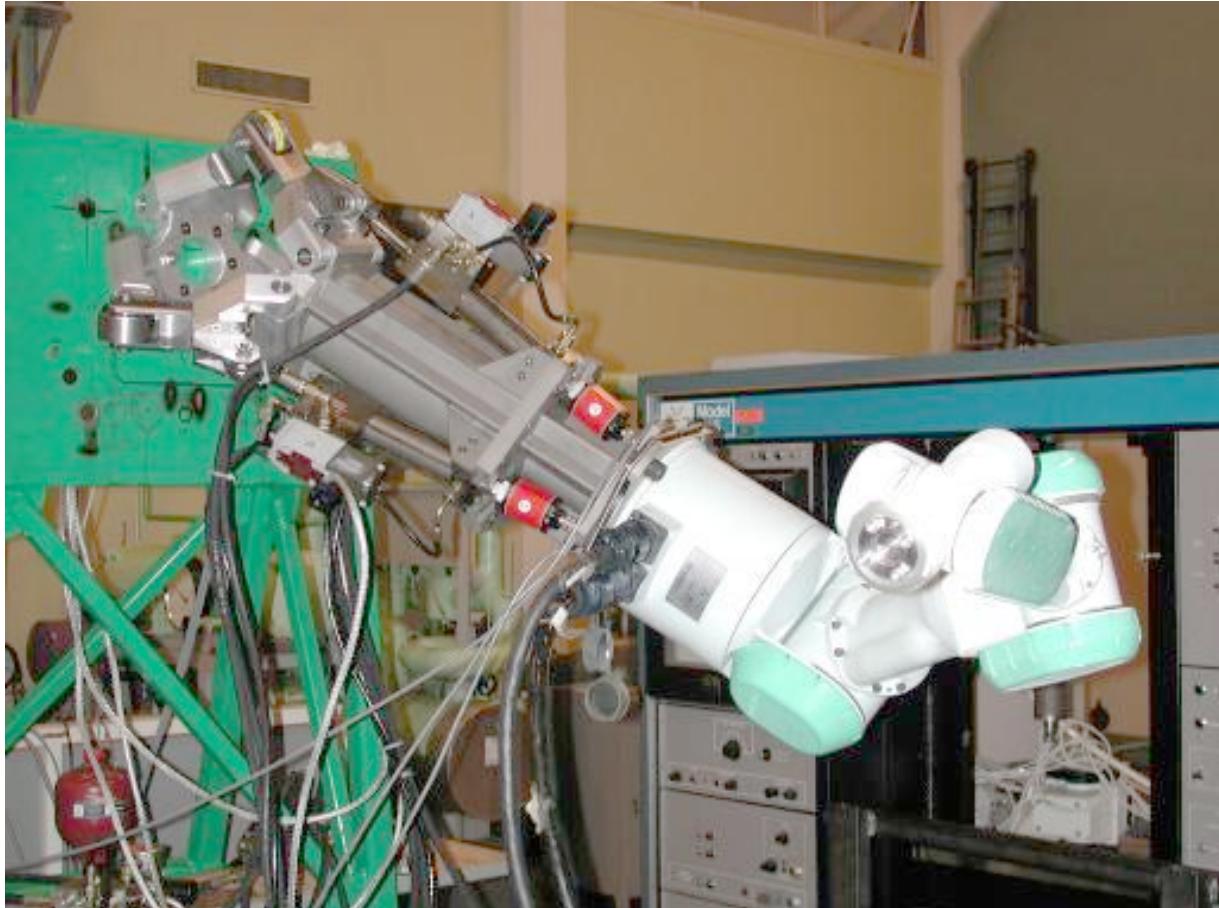


Modular control of industrial mechanics

Anders Stengaard Sørensen

Ph.D. Dissertation



The Maersk Mc-Kinney Moller Institute for Production Technology
University of Southern Denmark

1st April 2003

Contents

Foreword	7
Project course	10
Project and report structure	11
1 Introduction	13
2 Robots and modules	17
2.1 A layered reference model	17
2.2 A typical industrial robot	22
2.3 Advanced motion planning	24
2.4 Aggregated robots	26
2.5 Customized controllers	27
2.6 Modular robots	31
2.7 Conclusion	33
3 System architecture	35
3.1 Overall architecture	35
3.2 Generic embedded control nodes (GEECON)	36
3.3 Central control node	39
3.4 Information flow	41
3.5 The execution layer	48
3.6 Conclusion	54

4 Embedded platform	57
4.1 Physical demands	57
4.2 Computer platform	59
4.3 Architecture	61
4.4 Experience	64
4.5 Conclusion	65
5 Generic I/O	67
5.1 The fundamentals of I/O	68
5.2 Reconfigurable technology	73
5.3 Hardware platform	78
5.4 Internal architecture	79
5.5 I/O modules	82
5.6 The CPU interface	83
5.7 Conclusion	85
5.8 Future work	85
6 Network	87
6.1 Overall network demands	87
6.2 Network candidates	88
6.3 IEEE-1394 — Firewire	90
6.4 ARC-net	94
6.5 Discussion	96
6.6 Conclusion	96
6.7 Future work	97
7 DT-VGT interface	99
7.1 The DT-VGT platform	99
7.2 Switched mode valve amplifiers	101
7.3 Power supply	103

7.4	Daughter board	103
7.5	FPGA configuration	105
7.6	Software	107
7.7	Hydraulic control	110
7.8	Mechanical integration	112
7.9	Experience	113
7.10	Conclusion	113
8	PA-10 interface	115
8.1	Description	115
8.2	Observations	116
8.3	ARC-net interface	117
8.4	Software	117
8.5	Experience	118
8.6	Conclusion	118
8.7	Future work	118
9	System integration	119
9.1	Central controller	120
9.2	Painting a wheel barrow	120
9.3	The DockWelder system	123
9.4	Conclusion	124
9.5	Future work	124
10	Experience and discussion	125
10.1	Generic controller modules	125
10.2	Aggregated modules	127
10.3	The DT-VGT	127
10.4	Generic I/O	128
10.5	Network technology	128

10.6 Reference model	129
10.7 Hardware / Software co-design	129
11 Conclusion	131
12 Future work	135
12.1 Generic embedded control nodes (GEECONs)	135
12.2 Network	136
12.3 Central controller	136
12.4 Reconfigurable hardware	136
I Appendix	137
A Danish summary	139
B Published papers	143
B.1 Design of Double-Octrahedral VGT Manipulators	144
B.2 A development of parallel robotic modules for long reach applications	145
B.3 Towards the industrial usage of parallel kinematic modules in a fully modular robotic manipulator	146
B.4 A development of modular robots for flexible robotic manufacturing units	147
C Goal analysis	149
C.1 Overall task description	149
C.2 Conceptual basis	150
C.3 Goals	151
C.4 Criterions for success	153
D Work Breakdown Structure Analysis	157
D.1 Functional structure	158
D.2 Systems structure	159

D.3	Project Phases	160
D.4	Areas of effort	161
D.5	Work Breakdown Structure	163
D.6	Task descriptions	166
T	Technical	167
B	Business	174
O	Organisational	176
P	Political	179
M	Management	179
Bibliography		182

Foreword

The foundation for this project was laid by Odense Steel Shipyard (OSS), Institute for mathematics and computer science (IMADA) at University of Southern Denmark (SDU), and the company AMROSE A/S, as their quest to automate shipbuilding processes have led to the development of several robots, based on the aggregation of different mechanical modules.

At the same time, research in Applied Mathematics (AM) at University of Southern Denmark (SDU), have provided the community with a generic motion planner that can generate collision free motion paths for very complex aggregated robots moving in complex environments, like the interior of ship sections.

In order to expand robot applications in shipbuilding, several more or less experimental mechanical modules have been developed by OSS and AMROSE, in order to provide existing robots with additional horizontal reach and agility. Among these are:

- Telescopic modules — allowing horizontal reach.
- *Elbow* modules — able to bend in the horizontal plane.
- Variable geometry trusses (VGT's) — parallel kinematic modules.

By aggregating modules like this with conventional robotic mechanics, the reach and agility of conventional robots can be dramatically extended, and horizontal access to e.g. the interior of ship sections become a feasible alternative where vertical access from above is not possible.

When mechanical modules are aggregated into a complex mechanism, the aggregated modules essentially represents the mechanical parts of a modular robot system, but our reliance on conventional centralized controller technology, rob us of many of the benefits present in a truly modular system, and the work needed for systems integration and configuration remains practically the same as for a non modular system.

As one of the major barriers toward increased use of robots is in fact the cost of systems integration and configuration, rather than the cost of the equipment, we find it very interesting to overlay the existing modular mechanics with a corresponding modular controller technology, in order to simplify systems integration and configuration.

As we wish to contribute to the promising local development of Variable Truss Geometry (VGT) modules, this project uses the VGT modules as primary example and demonstrator, but will also

address other types of modules used by the local robotics community.

Project course

The project was originally meant to be the first in a string of semi parallel Ph.D. and master thesis projects at the Mærsk Mc-Kinney Møller Institute for Production Technology (MIP), related to the use of parallel kinematic modules in aggregated robots. In that context, this project should provide the controller platforms and infrastructure for experimental projects related to kinematics, motion planning, actuator modelling, control and similar high and low level controller issues.

Due to external events, the project environment quickly became too turbulent to support a complicated set of supporting projects:

- Odense Steel Shipyard decided to reduce their robotics development department dramatically over a series of layoffs, eventually abolishing the department.
- MIP changed their research priorities, reducing the budget for modular controller development to 15%, and cancelling the plans to employ relevant technical personnel.
- As AMROSE A/S relied heavily on commissions from OSS robotics department, AMROSE A/S was unable to survive the OSS cutbacks, and was eventually terminated¹.
- Potential Ph.D. students declined to engage in the proposed parallel projects, due to the changed research priorities and the turbulent circumstances.

In order to secure the necessary funding, we have integrated our project with the EU project *DockWelder*, where we are committed to develop a modular control system for a demonstration of aggregated robots used for arc-welding inside closed ship sections.

The high level of uncertainty surrounding the project combined with our commitment to produce practical results for the DockWelder project, led us to revert to a cautious and reactive project model, allowing us to adapt to a changing project environment and benefit from unforeseen opportunities.

¹It was later reconstructed in a very reduced form, as AMROSE Robotics ApS.

Rather than beginning with specifying the entire system, implementing a string of subsystems and integrating them in the end, it has been necessary to begin with a sketch of the overall system that is revisited and revised for each subsystem that is implemented, and for each relevant change in the project environment.

Following the cautious project model at the expense of efficiency and ambition, have allowed the project to survive and yield useful and interesting results in spite of a project environment that have changed beyond recognition.

By continuously adapting to the changing environment, opportunities and budget, we have managed to complete this project despite the grave difficulties imposed by external events.

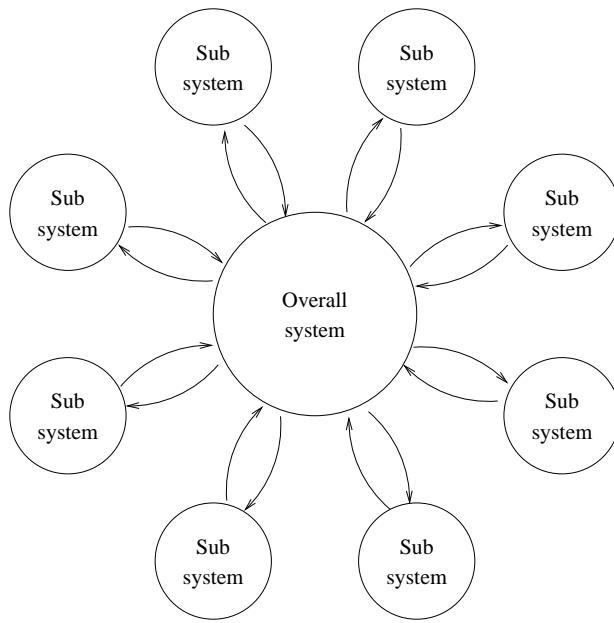


Figure 1: Cautious project model

Project and report structure

Using analysis techniques from project management [Riis98] & [Nicholas94], we have broken the project down into sub projects and tasks. Appendix D gives a thorough description of the Work Breakdown Structure (WBS) used throughout the project. The WBS have proved extremely useful for handling the sub projects and tasks in this project, and for keeping track of files and documentation. All subtasks are uniquely identified by their appropriate WBS code, and design files, source code etc. associated with each task can be found on the accompanying CD-ROM, using the WBS code of the task. The project breakdown is motivated by the goal analysis or task formulation provided in appendix C. The goal analysis is part of a situation analysis, that also includes a history analysis, partner analysis and analysis of the project environment. These have not been included as their results are either included implicitly or found irrelevant to this report. All the analysis are designated: WBS: M-01 and can be found in the corresponding location in the file hierarchy. For the convenience of the user, the material of M-01 have been indexed by a HTML document placed at: M/M-01/index.html

While the WBS described in appendix D might be used to structure our report, we find that our work have been quite unevenly distributed over the originally defined sub projects, and that some of it is not relevant to the report. Consequently, we have chosen to concentrate on the main technical issues, and have simplified the report structure with respect to the WBS.

Comparing the WBS of Appendix D, with the report structure of figure 2 reveals a simple map-

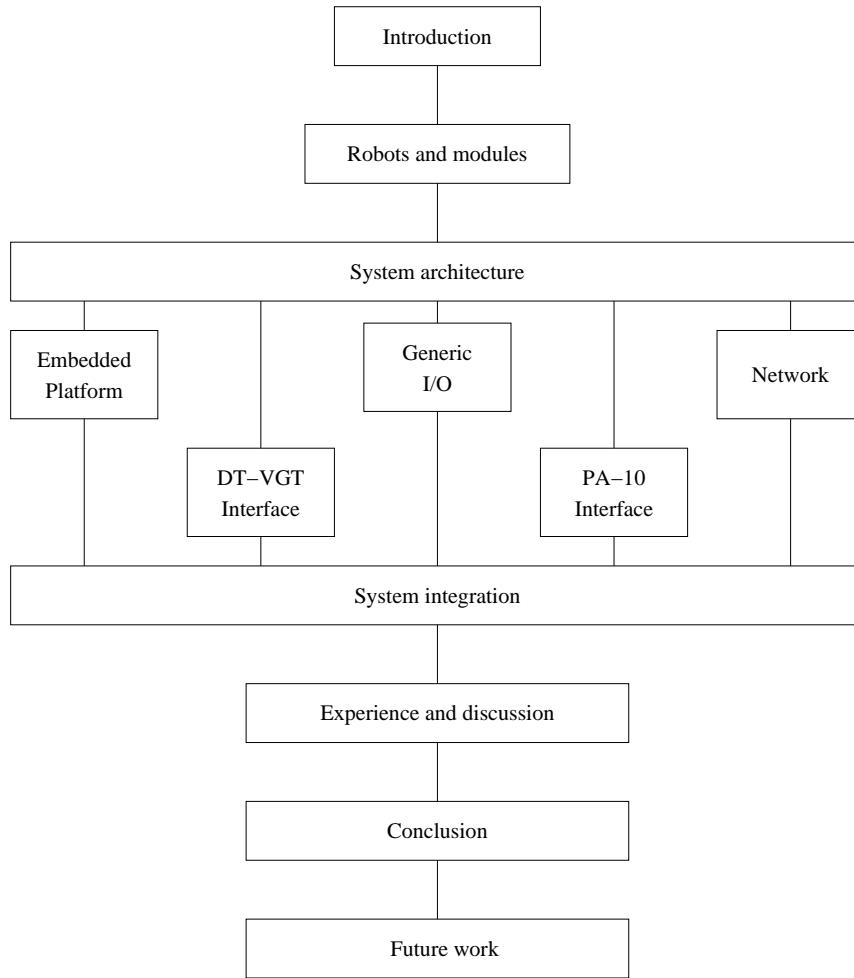


Figure 2: Overall structure of the report

ping from the original WBS to the report structure, testifying to the usefulness of thorough analysis prior to projects of this magnitude.

Chapter 1

Introduction

Modular mechanics have proved to be the most feasible way to design and construct highly agile robotic systems with inordinate degrees of freedom. As the local robotics community of Odense is dedicated to the control of such complex manipulators, modular robotics have been pursued by the community as an important byway. The experience of aggregating heterogeneous mechanical devices into complex robotic systems, have given rise to a vision of a robot system based on a limited number of heterogeneous *intelligent* mechanical modules, that can easily be assembled into an unlimited set of different robotic systems with a minimum of systems integration and configuration.

If such a technology came into common use, it would dramatically reduce the cost and work required to design and implement robotic production facilities, as one of the primary expenses of installing robotic systems is currently associated with systems integration and configuration.

Pursuing this vision — making it easy to assemble and configure different robots from a set of intelligent mechanical modules — several local projects have featured various forms of controllers embedded into the mechanical modules. These projects have shown the benefits of such an approach, but have not come up with solutions that can be used beyond limited laboratory demonstrations, as the implementation and integration of the embedded controllers invariably became too crude for practical use.

In this project we seek to develop a practical technology capable of transforming any commercial or custom build mechanical device into a self contained, automatically configurable *intelligent* module, that can be used as a building block in complex aggregated production system, in a *plug and play* fashion.

Our prior experience with aggregated robot systems indicate that one of the main barriers for implementing such a system is the lack of sufficiently flexible embedded controllers, that can be integrated with current and future mechanical modules. Commercially available embedded computer systems have proven to be either too large, limited or inflexible to be used as a generic embedded controller.

The main goal of this project is the development of a *generic embedded control node* — abbreviated GEECON — that can interface to any actuated mechanical device, perform the required low level control, and present the device in a way that enables it to be integrated into a modular robot architecture using only a communications network. In order to do so, we propose, investigate, implement and integrate a number of key technologies, related to GEECONs and their practical applications.

In order to be of practical value, the GEECON must be small, flexible and powerful enough to:

- Be embedded into the relevant mechanical modules.
- Be efficiently and easily interfaced to the relevant modules.
- Perform satisfactory low level control of the relevant mechanical modules.

To ensure that our GEECONs can be used with the widest possible range of mechanical modules, we present relevant assumptions and estimates for the design parameters, and aim to demonstrate the developed GEECONs with two robotic modules. We have chosen mechanical modules that represent very different aspects of robotics, ranging from parallel to sequential kinematics, linear-hydraulic to rotary-electrical actuators and custom experimental to commercially available mechanics.

This project is partly financed by the EU project *DockWelder*, where the GEECONs represent our contribution. The commitment to deliver working prototypes for DockWelder, calls for a practical approach and emphasizes the engineering aspects of our work.

Related work

In order to appreciate the relevance of related work, it is necessary to recognize that the popular notion of robot *control* is in reality a wide spectrum, ranging from low level feedback control of e.g. actuators, to abstract high level planning of the robots work. In chapter 2, we define a reference model that will help to distinguish the relevant aspects of a robot system, and facilitate a more detailed discussion of related work, than the cursory overview given below.

When surveying the area of *Generic robot control*, one is immediately impressed with the accumulated amount of work. A closer look reveal that the bulk of this work is concerned with the *planning* rather than the *execution* aspects of control, which is at the heart of this project.

Much of the work that touches on *execution* aspects, is concerned with software architecture and software design of generic robot controllers like [OROCOS]. Though execution aspects are considered, examples of practical applications are scarce. In our survey, we have only been able to find a few examples of *generic robot controllers*, that actually interface directly to a robot. In chapter 2 we describe the two most relevant: The *Open modular controller* (OMC) and *GENERIS*.

The large availability of embedded computer platforms make the use of embedded control a matter of course for most modular robot systems, for the same reasons invoked by this project. Typical examples of the use of embedded controllers are: [Dalgaard01] [CMU-arm] and [Powercube]. The main virtue of embedded controllers is their compactness, which is achieved at the cost of flexibility. Though an embedded controller can be designed for virtually any application, no single embedded controller can be used for all applications.

In our experience, very few attempts have been made at developing a robot controller technology that is both general and embedded at the same time, and no such technology seems to be readily available. OMC and GENERIS both have an inherent potential for embedded implementations, but as we will discuss in chapter 2 and 4 they rely on computer platforms which are too bulky to be considered *embedded* in our context.

Chapter 2

Robots and modules

There are many different aspects to modularity, and it can exist on all levels of a robotic system, from the mechanics, over the electronics to the software.

In order to be able to discuss and compare various systems and technologies, we begin by introducing a simple layered reference model, that can be used as a common frame of reference for describing and comparing various parts of a robot system. We then apply the reference model to a typical commercial robot, and go on using the reference model to describe the aggregated robot systems, developed and used at Odense Steel Shipyard. Throughout our descriptions we will emphasize on the modularity of existing technology.

2.1 A layered reference model

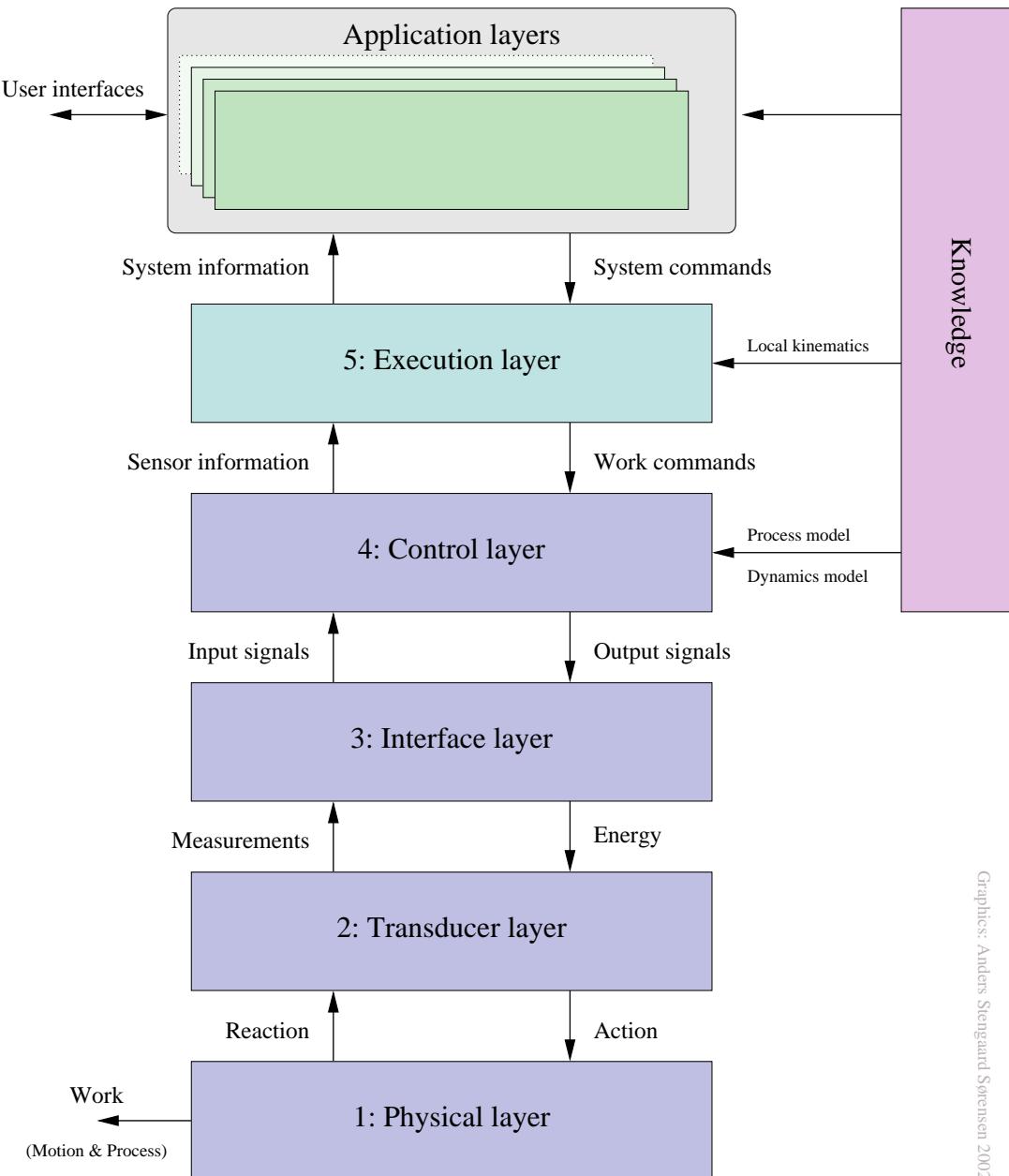
In order to enable comparison of various methods and technologies within robotics, we have defined a simple layered reference model to identify the relative location of components within a robotic system.

The model is inspired by the well known OSI reference model for computer networks[OSI], but also from robotic research such as: [Albus98]

Our model operates with 5 fixed lower layers, and a number of higher layers, referred to as *the application layers*. As we are not interested in the application layers in this context, we simply regard them as a sixth *meta* layer to our model.

We stress that this is a reference model, intended to aid description and comparison of robots and their low level controllers. It is neither a design nor implementation model.

Below we describe the individual layers, giving relevant examples.



Graphics: Anders Stengaard Sørensen 2002

Figure 2.1: Overview of the layered model

Physical layer

In this context, we define the work of a robot as the combined motion and process.

The physical layer is the part that actually perform *work*, typically the mechanical body and tool of the robot, along with the physical world that surrounds it, including the physical laws of nature.

The input to this layer is (mechanical) action, e.g. when an actuator is generating a force or a tool is performing a process.

The feedback from this layer is (mechanical) reaction, which can either be measured by sensors, influence the applied action or both.

Transducer layer

This layer represents the behaviour of actuators and sensors, which are *transducing* energy into action and reaction into measurements.

The linear relationship between current and torque in an electric motor, $\tau = k I$, is a simple example of an actuator that converts energy into action.

A typical example of a sensor is the angular encoder which converts angular movement into digital pulses which can be counted in order to determine the angular movement of a joint.

Interface layer

This layer is responsible for delivering and controlling the energy to the sensor-actuator layer, based on signals from the control layer.

A motor driver — or servo amplifier — along with a power supply, is a typical example of a component that converts a control signal into electric energy, e.g. a current.

An up/down counter that counts pulses from an angular encoder is a typical example of a sensor interface that converts sensor output to an adequate input signal for the control layer.

Power control components may contain some low level control loops, such as voltage- or current regulation in a motor driver or flow regulation in a servo valve.

Control layer

This layer is responsible for making the actuators and tools of the robot follow the commands given by the execution layer. It is also responsible for real-time sensor evaluation and for relaying sensor information to the execution layer.

This layer performs the low level control of individual actuators, using various control algorithms, where work commands are compared to measured values in order to compute an adequate output signal. PID control is a well known example of such low level control, but controlling complex mechanisms will often rely on more advanced control methods.

Although this layer is concerned with motion control, and the interface layer is concerned with power control, it may be hard to distinguish between control loops for power control and motion control in practical applications. This is especially true where such loops are not clearly nested within each other.

This layer can also perform *sensor fusion*, where a number of separate measurements are used to calculate a higher level of information. An example of sensor fusion is a *welding sensor* where measurements of the joint positions and welding current are used with a kinematic model and a process model, to calculate a corrective position offset for the robot tool tip.

In order to ensure smooth and stable motion, it is important to ensure real-time operation of the control layer. This is typically done by implementing the functions of the control layer on a computer with a real-time operating system or by using dedicated analog or digital hardware.

As this layer must run in real-time, it is important that the work commands from the execution layer is present when needed — typically at fixed intervals. This can be ensured in two ways:

1. Demanding real-time performance from the execution layer.
2. Buffering commands from the execution layer, allowing the execution layer to have non real-time performance, by letting it run ahead of the control layer.

Execution layer

The execution layer provides work commands to, and reads sensor values from the control layer. Typically, work commands are joint positions along with tool commands.

A typical execution layer function is interpolation and buffering, where the execution layer generates real-time work commands at a high rate for the control layer, based on non real-time work commands from the application layer.

The execution layer can also be responsible for online control of the process. For instance, it can change the joint positions slightly, in order to perform offset compensation of the robot tool, based on sensor information about the process.

Application layers

Although most of the functionality of a robot system belongs to these layers, they will only be treated superficially in this work, as most of the functionality here relates to motion planning, teach in, manual control, user interface etc. which is not in the scope of our work.

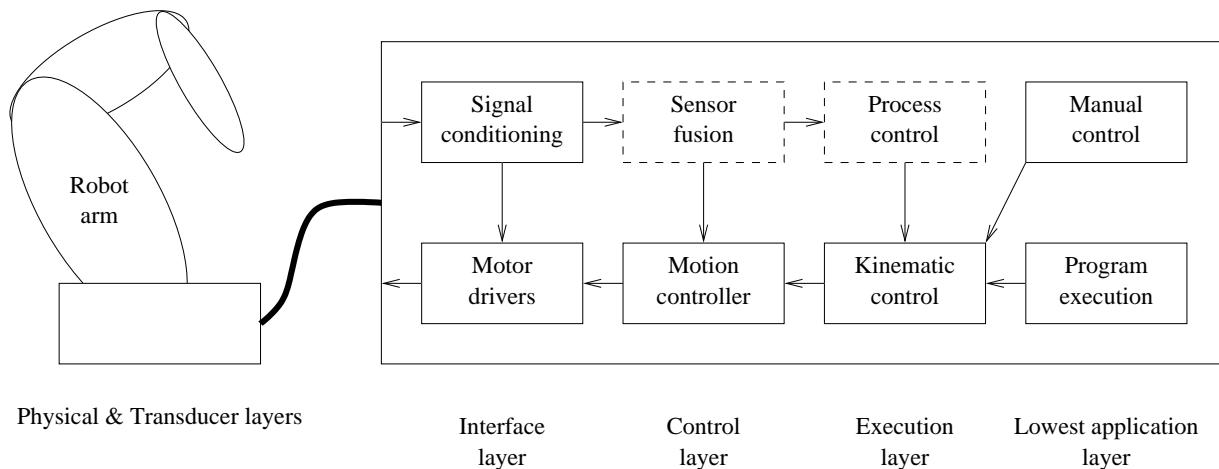


Figure 2.2: A typical robot/controller system

Basically, the responsibility of the application layers is to provide work commands to the rest of the controller.

A very simple example of such an application, is a program that will read a joint-file, generated by e.g. a motion planner, and relay the joint vectors in the file to the execution layer.

Another example could be a graphical interface, that continuously shows an animated 3-d model of the robot and it's status, based on sensor information.

As there are space for many application layers in this incomplete model, it is possible to extend the model to include a wide range of useful applications for robot control, including motion planning.

Knowledge

Knowledge of the robot system plays an important role in any robot control system. In practice this knowledge is often embedded in the functions of the various layers, but we choose to represent the sum of all knowledge about the system as a whole.

The knowledge can range from indirect knowledge about the robot dynamics, e.g. represented as control parameters in a PID controller, to exact CAD models of the robot and it's environment, used by a high level application, such as a motion planner.

2.2 A typical industrial robot

The arm

The robot arm itself may in fact be assembled from more or less modular components, but from the user/customer point of view, it is an atomic integrated unit. A typical robot arm consist of:

Mechanical superstructure clearly belongs in the physical layer.

Gears and transmissions also belongs in the physical layer. Some robots are build with standard gears, while others are designed with specialized and highly integrated gears.

Motors belongs to both the physical layer (mass, bulk, shape) and transducer layer (force, torque, current). As with gears, typical robots can have both standard and specialized motors.

Sensors are invariably used to measure joint positions, but may also be used to measure other relevant properties, such as joint speed or force. Like motors, sensors belong in both layers, and can appear both as discrete/standard or specialized/integrated sensors.

Motor drivers

Each motor has a separate motor driver, that channels a controlled amount of energy from a common power supply to the motor, which places the driver in the interface layer.

As the current through a motor is practically equivalent to the force/torque yield, while it is easy to measure and control, practical motor drivers are almost invariably designed/chosen to control the motor current/torque in accordance with an external set-point, given in analog or digital form. This calls for a simple control loop with a typical bandwidth up to a few kHz, which is either implemented in hardware, or with a dedicated embedded computer.

As motor drivers are specific to the type, size and specifications of a given motor, have simple external interfaces, and requires close integration between power and control components, they are usually implemented and regarded as individual *black box* components or modules.

In order to reduce mass and bulk of the arm, the motor drivers are separated from the arm itself, which makes bulky cables and connectors between the arm and the power electronics an annoying necessity. Usually the drivers are placed in a rack, along with the other electronics of the robot controller.

Displacing the motor drivers from the arm, means that the physical properties (mass, bulk etc.) of the drivers can be ignored with respect to our reference model.

Signal conditioning

An amount of signal conditioning electronics is necessary to convert the sensor measurements into usable signals. Some of this may be integrated with sensors inside the arm, and some of it may be placed in the controller rack, along with the motor drivers, but it all belongs to the interface layer.

Motion control

The motion controller of a robot, is supposed to ensure that each motor in the arm, is always sufficiently close to its intended position to allow the robot to perform its task. This places it in the control layer, and involves at least:

- A set-point for each motor, allowed to change dynamically within certain limits.
- Position feedback from each motor.
- A control algorithm involving explicit or implicit knowledge about the robot arm.

There may be more than one dimension to the motor set-point and feedback than position, e.g. velocity and torque, and the control algorithm and knowledge may range from the simple to the advanced. It all depends on the demands for the speed and precision of the particular robot.

The motion control can be implemented in analog or digital hardware, as well as in software, usually running on a dedicated real-time system. It is also quite common to make a mixed SW/HW implementation of the motion control.

Kinematic control

In order to position and move the robot tool in Cartesian space, robot controllers have an inverse kinematic algorithm that transform set-points given in Cartesian tool coordinates into set-points for the individual motors, allowing the robot to be controlled and programmed in Cartesian space.

We locate the kinematic control in the execution layer. It is invariably implemented in software, drawing on knowledge about the robots kinematic properties. If demands for speed or precision are high, it may also include other knowledge about the robot, such as dynamic and static properties, to compensate for deflections due to acceleration or gravity. It will typically run on a dedicated computer closely integrated with the motion controller, or possibly on the same computer as the motion controller.

Tools and process control

With respect to the physical, transducer and interface layers, a robot tool will simply add mechanical and electronic components which are equivalent to the components used for the robot itself.

The control layer may be expanded with functionality to fuse sensor data from the tool with sensor data from the arm, to obtain relevant process data.

The execution layer may be expanded with functionality for process control, applying Cartesian offsets to the arm, in order to control or adjust the process parameters.

Robot control and programming

A typical robot controller will include at least manual control, and the ability to execute predefined robot programs. The predefined programs may either be created by *teach in* using manual control, or they may be generated with more or less sophisticated off-line programming tools.

We place both manual control and program execution in the lowest of the application layers, as they both provide the execution layer with a dynamic set-point.

External axes on commercial controllers

As it is quite common to use external axes to expand the work area of a robot system, some commercial robot controllers can be used with external axes. Some have the ability to control a few axes themselves, others allow external control of the external axes and simply need to be kept informed of the position and direction of the robot base.

If a commercial controller can control or interface to external axes at all, the dynamic and kinematic control of the native and external axes are heavily decoupled, either by freezing the external axes while the native robot is working, or by using slow movements of the external axes to continuously keep the base of the native robot at an advantageous position.

2.3 Advanced motion planning

The art of mapping a tool configuration (combined tool position and pose) into a set of joint positions can be quite tricky, as it will typically involve singularities with multiple or infinite solutions. Mapping a tool path into a corresponding joint path is even more difficult, as the singularities must be negotiated gracefully in order to allow smooth movements. If the robot has

more degrees of freedom (DOF) than needed to describe the tool configuration¹, the robot is said to have redundant DOF's or simply to be *redundant*.

Redundant degrees of freedom cause an infinite number of solutions to the tool to joint mapping, causing a massive problem for traditional robot programming tools, that focus only on the tool configuration.

Most industrial robots have 5 or 6 DOF's so redundancy is often a regrettable by product when the work area of a robot is expanded by mounting it on some sort of positioning device like a gantry crane. The redundancy is one of the main reasons to keep the control of external axes decoupled from the native axes of the robot as mentioned above.

In production processes like ship building, the degree of automation can be increased dramatically if the robots can move inside structures, utilizing redundant DOF's to gain the flexibility needed to operate in narrow but well defined environments.

The company AMROSE A/S have succeeded in creating a generic motion planner based on constraint dynamics, that is able to create a collision free path for any mechanism, regardless of the number of degrees of freedom. The AMROSE motion planner uses a CAD model of the environment, a CAD description of the tool path (process) and a computer model of the robot describing its kinematic properties as constraints. Introducing artificial potential fields into the models, the motion planner is able to calculate valid collision free paths, expressed as a equidistant string of joint space coordinates mixed with process commands.

AMROSE A/S is a spin off company of the motion planning research, that have been a key area in Odense for more than a decade, and the AMROSE motion planner as well as various derivatives and other types of motion planners are at the disposal of the Odense robotics research community.

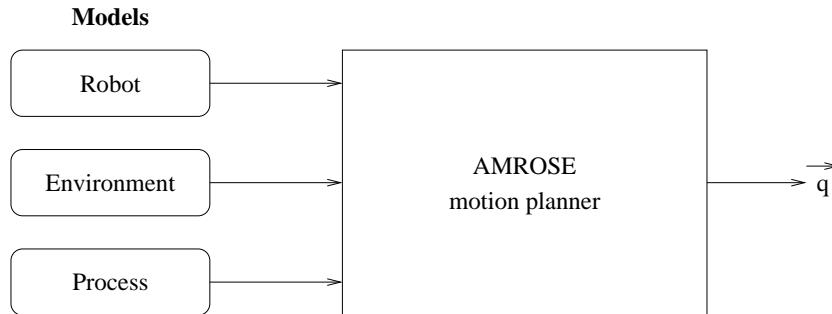


Figure 2.3: A global motion planner

Though the AMROSE motion planner has been demonstrated in real-time applications, dynamically adapting to a changing environment, it is only commercially available in an off line version. The commercial version is used to generate robot paths in advance, for later execution by a robot controller. The AMROSE motion planner have been demonstrated with 50 DOF simulated robots, and practical robots of up to 18 DOF.

The off-line AMROSE motion planner clearly resides in the higher application layers of our

¹The tool configuration can always be described in R^6 , but sometimes not all dimensions are relevant, e.g. tool rotation around own axis.



Figure 2.4: Custom build folding joint, to penetrate the inner labyrinths of a ship section

reference model, and it is only interesting to us for two reasons. The primary reason is that it makes it feasible to consider the practical use of advanced redundant robots. The secondary reason is that the use of constraint dynamics to describe the kinematic properties of a robot is highly modular, and can be used to facilitate self configuration of a robot system made from heterogeneous modules [Petersen01].

2.4 Aggregated robots

The easiest and most common way to add external degrees of freedom to a commercial robot arm, is to aggregate it with other commercially available mechanisms or units, such as linear displacement units, gantry cranes, turntables or even mobile platforms. In cases where commercially available technology does not suffice, mechanical designers tend to break their customized positioning systems down into manageable self contained units of a few degrees of freedom, like Odense Steel Shipyard (OSS) 3 DOF folding joint shown in figure 2.4

A common variation of the aggregated robot arise when the external axes are not applied to the robot arm, but are used to manipulate the workpiece in order to place it in an advantageous position for the robot tool. Work piece manipulators are also comprised of self contained mechanical units like turntables, X-Y tables, pan/tilt devices, hexapods etc. Commercially available devices are preferred, but for some applications custom build devices are used.

In some applications, more robot arms are cooperating to perform their tasks. Typically one robot is used as work piece manipulator, while the other manipulates a tool to perform a process, but there are an infinite number of possible applications for such setups.

Flexible automation

The massive costs to set up and configure robotic production facilities have previously made it infeasible to use robots for anything but large scale mass production. Both industry and researchers are striving to increase the flexibility of robot technology in order to make it feasible for small and medium sized production batches.

The quest to increase the flexibility of robot technology follow 3 basic paths:

- As CAD based motion planners like AMROSE have demonstrated, there is much flexibility to be gained from existing robot technology and installations, by exchanging traditional robot programming tools with automatic CAD based systems.
- Exchanging or supplementing existing controller technology, to allow increased use of external sensor feedback can also increase the flexibility of existing installations, as the robots will be able to recognize and react to differences in the processed objects as well as the environment.
- Increasing the modularity of robotic technology, to allow robotic installations to be designed, implemented, configured and reconfigured more easily.

Ultimately, the 3 paths will probably converge, as sensor feedback will sometime be able to replace and supplement predefined CAD models, and every part of a robot system will become a replaceable module with standardized interfaces.

The first path can be explored without major changes to the existing technology, as CAD based motion planners can be used off line, to generate predefined paths for existing robot installations and technology. All major robot manufactures now offer some sort of CAD based programming interface along with their robots.

The two second paths are hard to use with existing technology, as their requirements for the lower controller layers are different from what a typical industrial robot controller offers.

2.5 Customized controllers

There are a few valid reasons that might make it feasible to use a customized controller, rather than drawing on an existing commercial solution:

- It may be impossible to find a usable robot/controller system that supports the external axes (and process) needed.
- It may be necessary to utilize the added DOF's of external axes for agility as well as to enhance the workspace.
- Advanced applications may require more sensor feedback and evaluation than the native controller of a robot supports.

The first and third problem can often be solved by invoking a customized controller for the external axes or extra sensors, that can interface to a suitable commercial controller.

AMROSE have solved the second problem, by developing a generic motion planning method, that can calculate collision free trajectories for redundant robots operating in complex environments.

As the output of the motion planner is simply a stream of joint positions (and process states), it seems straight forward to simply feed the appropriate joint values to the appropriate execution layer components of whatever controllers are involved in controlling a robot system, but there are a few snags to this:

1. Few commercial robot controllers will accept programs in joint coordinates.
2. Controllers that accept joint coordinates, typically interpolates between adjacent points in joint space, making it necessary to keep the distance between adjacent joint coordinates quite small. This often result in *programs* becoming too large to be accepted by the program buffers of commercial controllers.
3. As practical versions of the AMROSE technology can only be used to generate off line programs, a process controller and kinematic controller are still necessary to apply on line compensation for small deviations between models and reality.

During the development and demonstration of the AMROSE system, many different ways to bypass these problems have developed, but only two alternatives have proved feasible:

- Using only robot systems where the native controller supports online kinematic compensation, while accepting programs in joint coordinates.
- Supplementing or replacing the native controller of commercial robots with a generic controller, especially developed to execute off line programs in joint coordinates on aggregated robots.

The OMC controller

As the first solution is far to restrictive to support the aggregated robots needed by Odense Steel Shipyard (OSS), OSS and AMROSE have attempted to developed a generic controller — known as the OMC or Open Modular Controller — that can replace or interface to commercial robot controllers, depending on the interfaces of the robot and controller in question. The OMC have been used successfully with the AMROSE motion planner in a number of applications.

OMC is designed as a framework that accepts hard- and software modules with specific interfaces, in order to arrive at a generic controller technology, using the power of modularity and reconfigurability. OMC has components ranging from the interface layer (hardware), to various levels in the application layers, including user interfaces, visualization, program execution etc. The framework is based on a combination of PCI/ISA, Windows NT®, and CORBA.

One of the key applications for OMC was the combination of process control and Cartesian control, illustrated in figure 2.5.

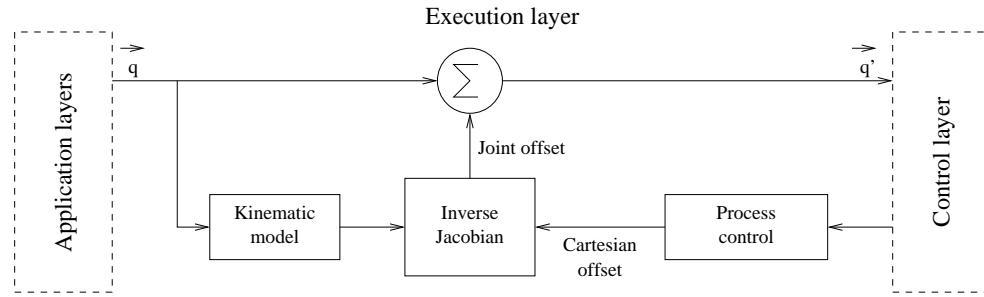


Figure 2.5: Process control in OMC

The modules for process evaluation and control of the arc-welding process, controls the welding process by applying small Cartesian offsets to the welding tool.

In stead of applying a complete inverse kinematic algorithm to control the robot in Cartesian co-ordinates, OMC relied on an inverse Jacobian to map small Cartesian offsets into corresponding offsets in joint space. The inverse Jacobian is continuously generated from the forward kinematic model of the system.

Figure 2.6 shows a typical application of the OMC with an aggregated robot.

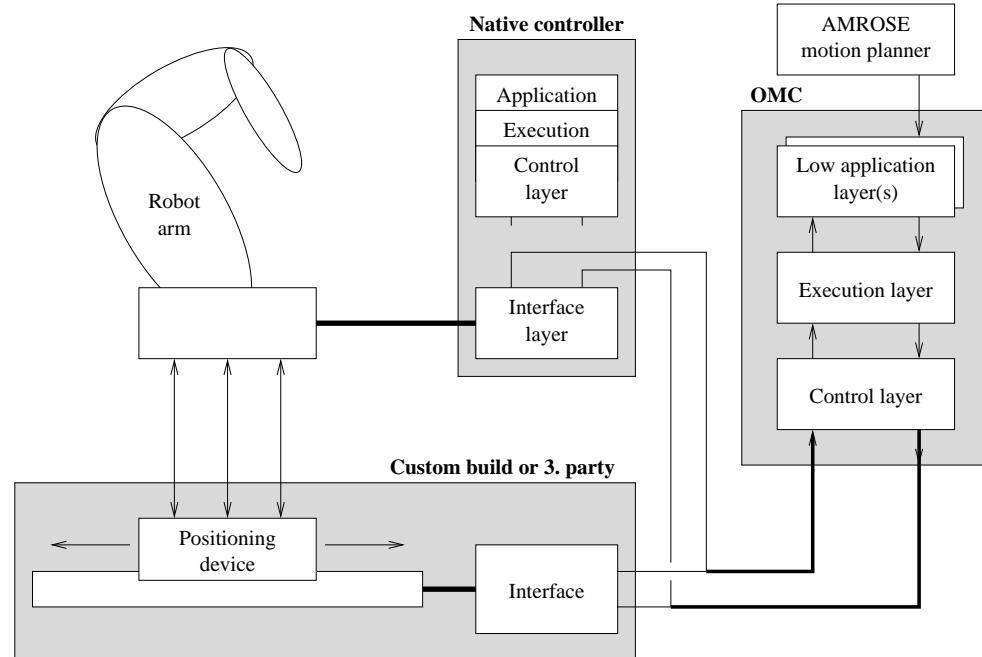


Figure 2.6: A typical OMC application

Note how the OMC replaces the higher layers of the native arm controller by literally cutting the original connections, and replacing them with connections to the control layer of the OMC. In some cases, the native controller would even be completely replaced by 3. party servo amplifiers. This wasteful practice is necessary as commercial robot vendors are reluctant to share the implementation details of their controllers that would allow an interface at a higher level.

The modular framework of the OMC was supposed to ensure that it could easily be interfaced and adapted to every conceivable mechanism and process, but the poor choice of platform technology proved fatal to this intention.

The use of a poorly documented non real-time operating system intended for desktop computers, made it virtually impossible to implement reliable control layer components in software, and the OMC came to rely on the PMAC motion controller PCI boards for motion control. The reliance on PMAC boards made it impossible to miniaturize the OMC hardware², and it was only a partial solution to the real-time problems of the control and execution layer components.

In spite of the original intentions of making a flexible distributed controller, the bulkiness of the OMC hardware prevents a close physical integration with the hardware it controls. The reliance on a specific PCI motion controller board limits it to control certain actuators (electric motors), and the difficulties involved in tricking Windows NT® to work in real-time applications make it excessively expensive to configure the system for new applications.

Toward the end of this project, the OMC partners finally terminated further OMC development, as the technology was deemed infeasible.

The GENERIS® controller

The European Commission Joint Research Center (EC-JRC) [EC-JRC] have developed a generic controller, called GENERIS®, that is very similar to the OMC, but have acknowledged the importance of real-time performance of the control and execution layer [GENERIS]

In comparison with the OMC, GENERIS draws a very sharp line between the application layers and the execution layer. Like the OMC, application layer SW is implemented under Windows, and can be distributed over a number of PC's known as GENERIS hosts.

Execution and control layer components are implemented on industrial computers under the highly accredited real-time operating system VxWorks. The computers implementing the execution/control layer are known as GENERIS Numerical Control or simply GNC.

The combination of execution and control layer allows each GNC computer to control one or more processing units — or robot cells —, composed of e.g. a robot arm, a robot tool and a positioning unit. The GNC architecture is tailored to standard modular industrial computer systems, such as VME, Compact PCI or PC-104, where a CPU module and a number of I/O modules reside in a centrally placed rack, connected to the individual sensors and actuators via

²By using PC-104 or similar miniaturized IBM-PC compatible technology

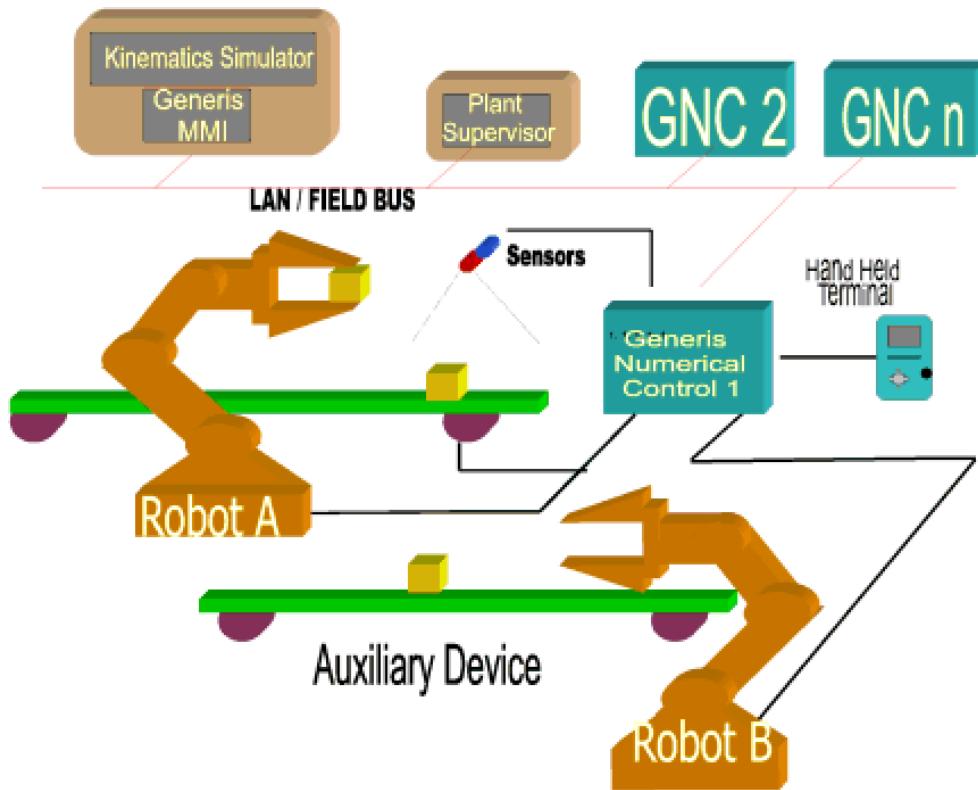


Figure 2.7: Overview of a GENERIS® system

a web of wires. As performance and wiring limits each GNC computer to control a few process units, more GNC computers can be connected to a GENERIS system, to allow an entire factory to be included into the system.

While the GENERIS architecture is well suited for production facilities with heterogeneous robot cells, including aggregated robots and custom designed mechanical units, sensors etc. it is obvious that the GNC's does not take full advantage of the modular nature of the mechanical modules that constitute a robot cell. A robot cell can not be reconfigured on the fly, as reconfiguration require changes to the wiring I/O hardware, and software of the GNC computer in charge of the cell.

2.6 Modular robots

As flexibility and reconfigurability are very obvious advantages of modular technology, various attempts have been made to create fully modular robots and robot components for experimental as well as commercial applications. Modular robots are dominated by two major trends: Commercial products where modularity is simply a means to achieve an effective solution, and research projects where modularity is an end in itself. A typical example, and a comprehensive

survey of the latter is given in [Pakpong01].

The attempts at modular robots vary greatly in aim, ambition, application and technology, but they have a number of aspects in common. The lowest common denominator of the field is the industry trend towards smart actuators and sensors, where the necessary interface layer components are physically integrated with the actuator or sensor in question, along with an embedded computer working primarily as a communications controller, connecting the intelligent unit to a common network or field bus. The embedded computer can also implement some simple control or even execution layer components, in the form of velocity or position control, interpolating trajectory generators etc.

In most cases, the various commercial suppliers respect existing, non modular, standards for the physical properties of their products, so they are mechanically compatible with previous non intelligent versions. Some attempts have however been made to market products where the modularity extends down to the physical layer as well. One of the most convincing examples of this is the *powercube* concept by *Amtech Robotics* [Powercube], illustrated in figure 2.8.

Amtech manufactures a small range of compatible intelligent PowerCube modules, for rotary and linear movement, as well as a number of gripping tools and special purpose modules, such as pan/tilt. Except from the power supply, All of the modules are completely self contained, and accept motion commands via a common CAN-BUS network.

Concepts like PowerCube are generic in principle, but in practice each of these concepts address a very narrow segment, with respect to power, speed, precision and cost. This type of modular approach make very nice components in an aggregated robot system, and is well suited to demonstrate the power of modularity, as shown in [Nanyang], but it is not sufficiently generic to cover an entire industrial robot cell.

Apart from issues related to mechanical diversity, and compatibility in general, one of the principal drawbacks for intelligent actuators are their lack of knowledge about their role in an aggregated mechanism. The overall kinematic controller as well as each intelligent actuator must still be configured manually in accordance with the static, dynamic and kinematic properties of the mechanism. As the dynamic and static properties of a robot can vary dramatically with the position of the robot joints, the parameters of intelligent actuators must be changed dynamically by a central intelligence in order to obtain optimal performance. This problem arises because only



Figure 2.8: A rotary PowerCube actuator

the actuators and not the links between the actuators are included under the modular controller.

In general, intelligent actuators and sensors makes the hardware configuration much easier, as individual signal wires are replaced by common networks, but the burden of configuring software and parameters remain largely the same as if using a pure central controller. Configuring modular robots is currently a popular research topic, but the bulk of this research seems to be preoccupied with various aspects of abstract or experimental *universal modules* rather than practical industrial technology.

Some research and development effort have gone into developing modular technology for specific mechanisms, such as serial robot arms. In such cases, the entire physical, transducer, interface and control layer is closely integrated within each module, making it possible to store all relevant information within each module, and configure the aggregated system in a plug and play fashion.

One of the most serious attempts at building a modular arm is performed by the Robotics and Mechatronics department of *Deutsches Zentrum für Luft- und Raumfahrt* (DLR) [DLR]. Over 3 generations of mechatronic development, they have succeeded in designing and manufacturing a powerful light weight, fully integrated module for robot arms, known as LBR-III. Each module has two degrees of freedom, configured as perpendicular rotary joints. The arms that can be assembled from the LBR-III modules have similar specifications as commercial arms of similar size, except that they are significantly lighter, as LBR-III was originally designed for space applications.

Currently DLR only has one type of arm module, and a small number of experimental tool modules, but it is quite possible that they may expand the selection of modules to expand the number of possible applications.



Figure 2.9: LBR-III modular arm

2.7 Conclusion

We are not aware of any attempts to develop a generic controller technology that can take full advantage of the inherent modularity of the traditional components of robotic systems. The nodes of current generic controllers aim at controlling an entire robot cell, rather than its individual parts. Current research and development of intelligent mechanical modules does not seem to

converge toward practical standards to support reconfigurable industrial robots.

As generic embedded controllers will mainly be useful in combination with a generic motion planner, we believe that the low availability of such motion planners is the key to explain the lack of interest in generic embedded controllers.

As the robotics community of Odense have access to a working generic motion planner, it seems natural for us to investigate generic embedded controllers as part of our research and development projects.

Chapter 3

System architecture

Abstract

During this chapter, we propose and elaborate on an overall architecture that will enable utilization of the modular nature of aggregated robots. We discuss various aspects of our proposal with respect to performance, bandwidth and other relevant requirements.

3.1 Overall architecture

In order to utilize the inherent modularity of modular mechanics, we propose an architecture that is based on small and simple, yet powerful and flexible embedded control modules or nodes, which we will refer to with the abbreviation GEECON for generic embedded controller node. Such control modules will make it feasible to integrate a GEECON with each mechanical module in a robot system, rather than one for each robot cell.

Integrating a GEECON into each mechanical module, can be viewed as transforming passive mechanical modules into active intelligent ones, which opens a number of interesting possibilities:

- Dramatic reduction in the system complexity, as signal cables to individual sensors and actuators are replaced by a common network.
- Easier construction, maintenance and reconfiguration of robot systems, due to the reduction of complexity.
- Increased compatibility between heterogeneous technologies, as the GEECONs can present heterogeneous mechanical modules in a homogeneous way to higher control layers.
- As intelligent mechanical modules can store relevant information about themselves, such as kinematic parameters, a network of such modules makes it possible to support automatic configuration of high level applications, such as motion planners and user interfaces.

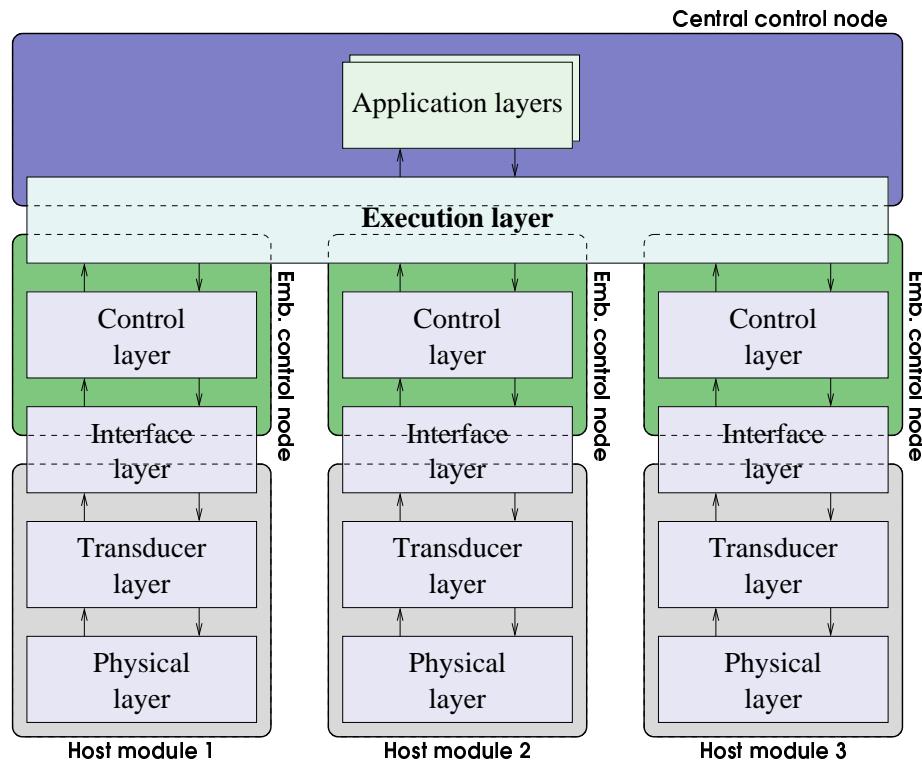


Figure 3.1: Distributed controller model

Although it would be interesting to study methods to distribute all components of a robot controller, we prefer to restrict the GEECONs to tasks that are entirely local to their host modules. All tasks and functions that requires an overview that involves more than one host module (global) will be performed by a central control node.

This architecture is illustrated in figure 3.1, where a distributed version of the layered reference model (hard corners) is superimposed on a layout of a central control node with a set of GEECONs and accompanying mechanical host modules (round corners). As indicated in the figure, the distinction between local and global tasks and functions for an aggregated robot system, implies that the components binding the distributed system together, resides in the execution layer. Execution layer components that use e.g. the kinematic model for the entire robot will thus be implemented on a central controller node, while execution layer components, such as interpolation at joint level, will be implemented on the corresponding GEECON.

3.2 Generic embedded control nodes (GEECON)

Considering the GEECONs and their relation to the host mechanical module, we propose to use the simple implementation and integration model shown in figure 3.2, where the interface

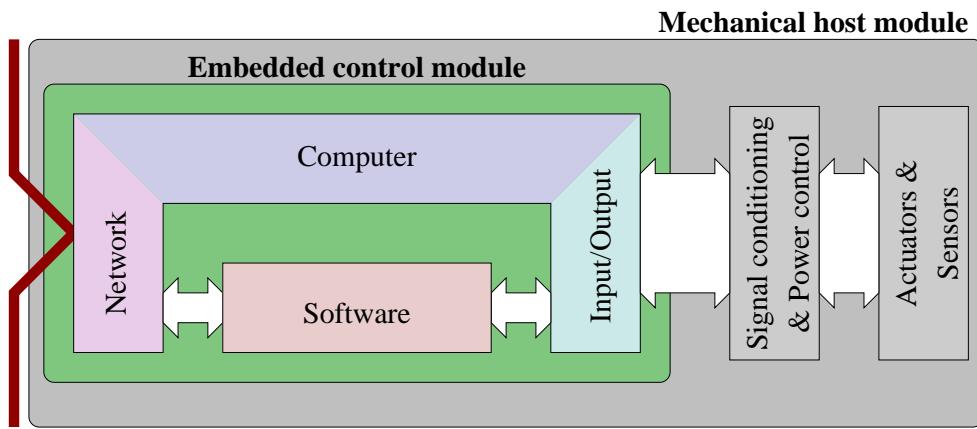


Figure 3.2: GEECON embedded in a host mechanical module

toward the central controller is defined by a network and the interface toward the mechanical host module is defined by the set of I/O signals necessary to control the host.

In addition to power supply and other necessary support systems, not shown in the figure, the hardware of the GEECON consists of computer, integrated with a network interface and an I/O interface. While the network interface must be identical or compatible for all the GEECONS in a system, the I/O interfaces may need to be quite different in order to suit the demands of different host modules.

Interface to host modules

As we want to be able to use our controller architecture with the widest possible array of different host modules, we can not assume much in advance, but must allow for extreme variations in the demands for connecting to and controlling mechanical host modules.

One assumption that we do make, is that each host module will be equipped with suitable power control and signal conditioning systems to allow a computer to interface to it through a set of I/O connections comprised of analog and/or digital electrical signals within common standards or practise. If this is not the case, we assume that we can implement such a system ourselves, external to the GEECON.

Although it may often be the case, we do not assume that a complete interface layer is implemented in the host module. On the contrary, we assume that it may be beneficial or even necessary for the GEECON to contribute to the signal conditioning and power control, with various forms of signal processing, which may be implemented in analog or digital I/O hardware, or in software.

The actuator/sensor systems on the DT-VGT modules we use as primary evaluation example in this project, is an excellent example of a host module with a partial interface layer, where we

have to supply power amplifiers for the actuators, and implement I/O hardware and software that participates actively in feedback systems to control the power flow (chapter 7).

In the other end of the spectrum, we might be confronted with host modules with native controllers that implement not only the interface layer, but possibly also the control- and even higher layers. In these cases, the role of the GEECONs is to interface to, and utilize, the native controller in the best possible way.

The PA-10 robot used as secondary evaluation example for this project is a typical example of such a setup, as the native controller of the PA-10 implements a full interface layer, and a partial control layer in form of velocity control of the individual actuators of the robot. The native controller has an open architecture that allows our GEECON to interface to the partially implemented control layer via a field bus in order to give velocity commands and receive position feedback (chapter 8).

Given the large variation of possible host modules, it is clear that we can not confine the interface between GEECON and host module to a single layer of the reference model, let alone define or describe a usable generic interface. We simply have to accept that the border separating the responsibility between GEECON and host module must be very fuzzy in order for the GEECONs to be general.

Interface to central control node

Regardless of how and where the control- and interface layer components are implemented, the result will enable the sensors and actuators of the mechanical host module to be read and controlled, by the software of the GEECON.

This ability will be common for all modules of the system, and it gives us the possibility of presenting a large array of heterogeneous robot modules in a homogeneous way to the central controller node.

Since the kinematic properties of the possible host modules can vary greatly, we believe that the most general way to present a host module, is as:

- A set of actuator systems, which movement follow the commands of the central controller node.
- A set of relevant feedback values that can be read by the central controller node.
- A database of knowledge, about the host module, such as kinematic and dynamic models, that can be accessed by the central controller node, and used for global control of the robot, such as kinematic compensation and motion planning.
- A state machine, that allows the central controller node to control and verify the state of the individual robot modules, for example: Uninitialized, Inactive, Active and Error.

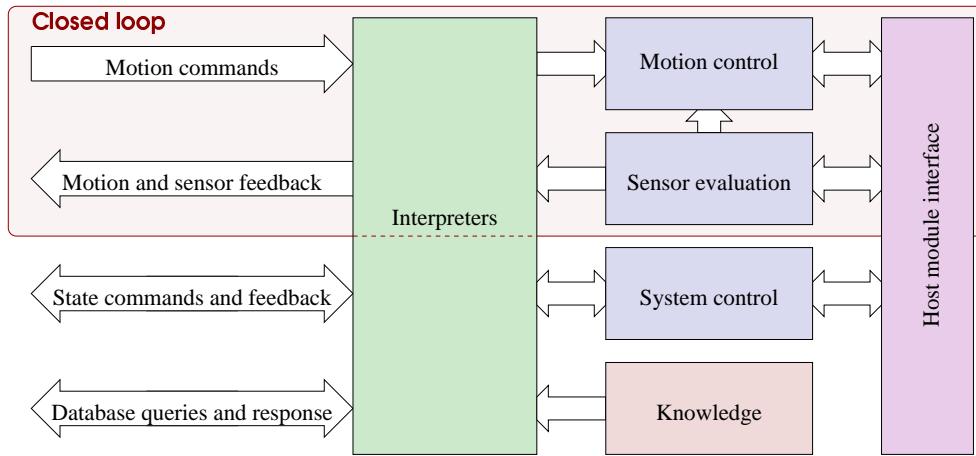


Figure 3.3: Sketch of the command structure of a GEECON

This way of representing different robotic modules will allow the GEECONs to have a homogeneous framework with regard to hardware platform, network, and high level software, which can support the application specific I/O, motion control, sensor evaluation and system control components, as well as relevant knowledge about the host mechanism.

Figure 3.3 sketches the command structure of such a representation, where we note that the components and information involved in specifying, controlling and measuring motion are all part of closed loops, which leads to requirements for the real time performance of communication and execution related to these components and information. We will return to discuss real-time requirements for the GEECONs when we discuss the information flow of the entire system.

3.3 Central control node

Figure 3.4 shows how the central controller node adds the remaining high level controller components to a modular robot system, where the low level layers are implemented on the individual robot modules.

The high level execution layer implements the part of the execution layer which require global knowledge of the robot, e.g kinematic compensation, or similar functions which are not practical to distribute to the GEECONs.

If a suitable abstraction layer, or module interface, is used to hide the implementation details of the distributed part of the system, it becomes possible to reuse existing execution layer software for kinematic compensation, process control etc. which in turn allows reuse of compatible application layer software, such as motion planners, visualisation and user interfaces.

The possibility to maintain compatibility with existing generic robot controllers, such as OMC and GENERIS, through implementing suitable abstraction layers, is very interesting, as it re-

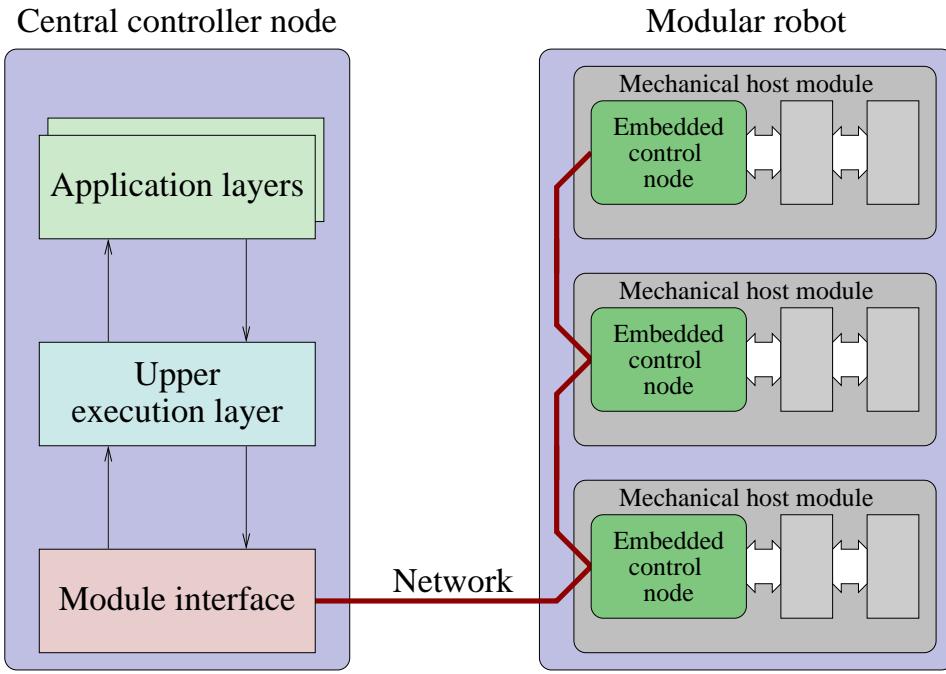


Figure 3.4: Central control node with a set of robot modules

leaves us of the task to implement the components of a complete controller, if we oblige a few prerequisites:

- The network technology used to connect to the modules must be compatible with the computer technology used for existing generic controllers.
- The protocol for communication with the GEECONs must be open, in order to allow others to implement suitable abstraction layers.

As execution layer and application layer components like process control, user interfaces and real-time motion planning, may need sensor feedback that is not directly related to the robot position and motion, it is important to acknowledge the requirements such feedback may impose on the communication network.

Apart from simple, one dimensional process feedback signals, with fairly low bandwidths¹, we have had very little experience with non motion related feedback. Apart from process feedback, future applications may call for proximity sensors, distance sensors, cameras, sonars, range scanners etc. etc. in an unknown number and in unknown combinations. Although GEECONs associated with the sensors in question may be able to reduce the amount of information to be transmitted to the central controller node, we are not able to divine which bandwidth and latency demands such sensors will put on the network.

As we are not able to offer any sensible limits to the future use of sensors, we must recognize that

¹typically below 100 Hz

we will never find a network technology that can ensure that our controller architecture allows generic use of sensors. We must settle for a compromise that allow us to reserve a sensible bandwidth for future sensor integration.

3.4 Information flow

The bandwidth and latency requirements for the information flow throughout a controller architecture has a profound effect on its implementation requirements. In this section, we will discuss the information flow of robot controllers, in order to get an understanding for the principles that govern the corresponding requirements.

Figure 3.5 sketches the primary information flow of a typical robot controller in terms of our layered reference model, which takes the form of a number of nested loops, due to the reliance of feedback. Static information about e.g. kinematic properties are left out, as it can be exchanged during initialization, and plays a minor role compared to the band width considerations of the closed loops.

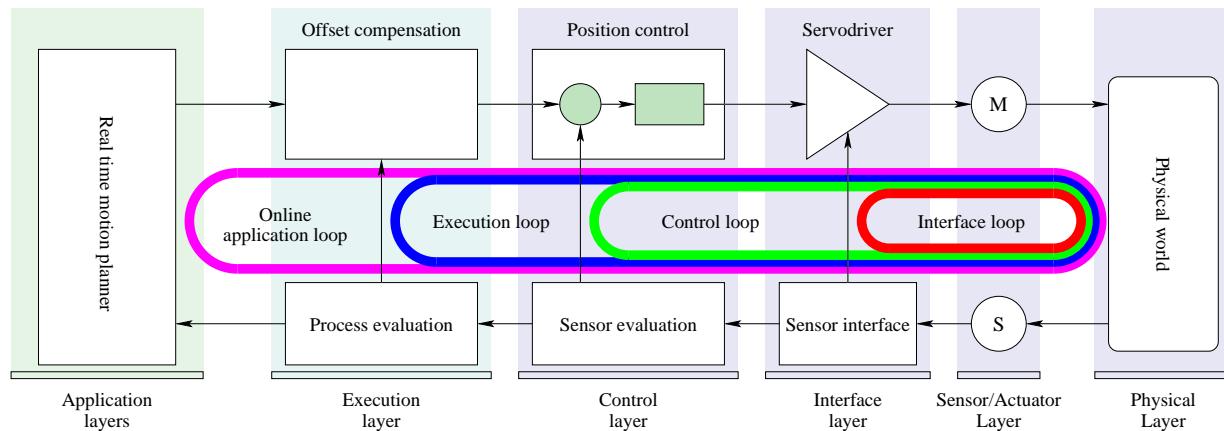


Figure 3.5: Nested loops in the layered model

In general, the layers close to the physical layer handles simple information with a high bandwidth, while the higher layers handles more complex information at a lower bandwidth. A servo amplifier of the interface layer may easily control motor current with bandwidths well above 10 kHz, while a kinematic controller of the execution layer may adjust the tool offset a few times a second.

In practice, the bandwidth of a given component or communication channel is chosen with respect to many factors, where we can emphasize: Physical necessities, physical limitations, need for synchronisation with other components, requirements for signal processing algorithms, availability of communication bandwidth and computing power.

Robot bandwidth

In order to appreciate the considerations of higher layers, it is instructive to begin by considering the physical capabilities of the mechanics and power control of a robot or other mechanical system. This discussion encompasses the entire physical and transducer layer, and many aspects of the interface layer. The bandwidths and sample frequencies discussed here applies to the *interface loop* as well as the *control loop*.

The variation of technologies and implementations used in industrial mechanics, makes it impossible to define metrics which are both generic and practical. The metrics defined in the following are useful for reference and comparison, but they do not necessarily take all aspects of practical mechanics into account.

In general, an actuator system of a robot or other industrial mechanism can be considered a transfer function, where the actuator converts the energy from the interface layer into a force, that accelerates the mass of the physical system. We can not assume much about the actuator system in terms of transfer functions, power supply or load, but we can safely assume the following:

- The acceleration will be limited due to a limited power supply and actuator efficiency.
- The resolution of feedback sensors are limited due to natural- and/or quantization noise.
- The requirements for actuator precision are limited, as actuator accuracy reflects the requirements for tool accuracy.

As the limits may indeed change dynamically as the robot moves, we are forced to contemplate only maximum acceleration, resolution and accuracy, as sketched in figure 3.6

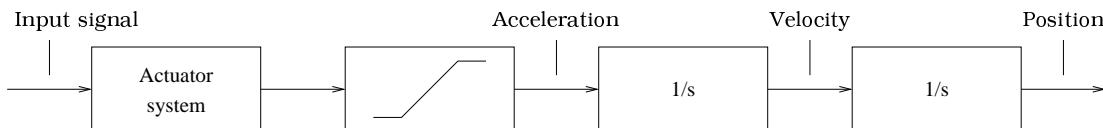


Figure 3.6: Worst case model of actuator system

We introduce the following metrics, that relates the input of the actuator system, to the position:

Detection bandwidth — f_e : The highest frequency of a sinusoidal input signal that will, in worst case, cause a motion that can just be detected with the feedback sensor.

Tolerance bandwidth — f_Δ : The highest frequency of a sinusoidal input signal that will, in worst case, cause a motion larger than the accepted position tolerance of the actuator system.

In our worst case model, the transfer function of the actuator system gives rise to a square wave acceleration, with amplitude equal to the highest possible acceleration: $A_{\ddot{x}} = |\ddot{x}|_{max}$, and a period: $T = 1/f$, where f is the frequency of the sinusoid in Hz. This leads to a triangular

velocity profile, with amplitude: $A_{\ddot{x}}$. And a piecewise parabolic position profile with amplitude A_x .

$$\begin{aligned} A_{\ddot{x}} &= |\ddot{x}|_{max} \\ A_{\dot{x}}(f) &= \frac{1}{4f} A_{\ddot{x}} \\ A_x(f) &= \frac{1}{32f^2} A_{\ddot{x}} \end{aligned} \quad (3.1)$$

It is clear that the amplitude of the movement will decrease steadily with increasing frequency. We define the detection and tolerance bandwidths:

$$\begin{aligned} f_\epsilon &= \{f \mid A_x(f) = \frac{N}{2}\} \\ f_\Delta &= \{f \mid A_x(f) = \frac{\Delta_x}{2}\} \end{aligned} \quad (3.2)$$

Where N is the amplitude of the noise or quantization level of the position feedback sensor, and Δ_x is the allowed tolerance for the actuator position. As we have made a worst case analysis, the bandwidths can be expressed as:

$$\begin{aligned} f_\epsilon &\leq \sqrt{\frac{A_{\ddot{x}}}{16N}} \\ f_\Delta &\leq \sqrt{\frac{A_{\ddot{x}}}{16\Delta_x}} \end{aligned} \quad (3.3)$$

In light of our worst case consideration, the individual actuator systems can be considered low pass filters, as illustrated in figure 3.7.

These filters can be thought of as the ultimate reconstruction filters of a sampled system, i.e. the robot controller, and the worst case band widths can assist us in choosing the internal sample frequencies of the controller.

If we choose a sample frequency of $f_s > 2f_\epsilon$ we are guaranteed that aliasing will not produce a measurable disturbance to the position of the system.

As long as the sampling frequency $f_s > 2f_\Delta$ disturbances caused by aliasing will be within the position tolerance. In any case, it must be kept in mind that disturbances caused by aliasing will be superimposed on other disturbances, if such exist.

As an example, we consider a linear actuator with load, that has a maximum acceleration of $|\ddot{x}| \leq 100m/s^2$, a range of $x \in [0; 0.5m]$, a feedback sensor with a resolution of $2\mu m$, equivalent

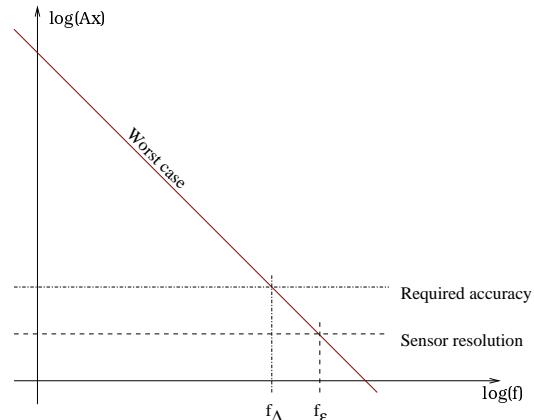


Figure 3.7: Sketch of bandwidths

to a S/N ratio of $108dB$ or 18 bits of resolution. The actuator is used in an application that calls for a position accuracy of $\Delta_x \leq 25\mu m$

In the context of industrial robots, this example is rather extreme, on account of acceleration as well as resolution, and we do not expect to encounter robots with such extreme demands in practical applications of our generic controller. The bandwidths for the example are:

$$f_e = \sqrt{\frac{100m/s^2}{16.2\mu m}} \simeq 1.8kHz$$

$$f_\Delta = \sqrt{\frac{100m/s^2}{16.25\mu m}} \simeq 500Hz$$

Which suggests that the robot controller should use a sample frequency between $1kHz$ and $3.6kHz$ to feed the actuator system with a sampled control signal.

Although this way to evaluate the bandwidth of an actuator system is rather coarse, we find it is a useful tool for initial evaluation and considerations regarding control systems in general.

In practice, our simple worst case estimate of bandwidth may be considerably higher than the real bandwidth of the actuator. Such discrepancies can be caused by various reasons, such as:

- The transfer function of the actuator have more poles than the double integration of our model, making it a more efficient low pass filter.
- The actuator speed is limited due to e.g. friction or back EMF in electric motors.
- The power controller have internal low pass filters that add to the characteristics of the actuator.

On the other hand, it might be beneficial or necessary to use a higher sample frequency than implied by the mechanical bandwidths. Major reasons for this are:

- Subsystems within the actuator system have bandwidths that are not limited by the mechanics of the actuator. Typical examples of this are: Voltage or current control loops for electric motors, pressure or flow control loops for hydraulic or pneumatic actuators.
- The signal processing of the controller can benefit from oversampling.
- Although aliasing will not have adverse influence on position control, it might lead to unwanted vibrations or even acoustic nuisances.
- A wish to synchronize with other sampled systems.

Interface loop bandwidth

It is quite common for subsystems within the interface layer to have bandwidths in excess of the mechanical bandwidths discussed above. It is not uncommon that the bandwidth of such

subsystems exceeds the mechanical bandwidth with several orders of magnitude. Preferably, these high bandwidth *interface loops* are implemented in the power control components native to the actuator, so the GEECON will primarily be concerned with the *control loop* and possibly higher loops. Typical examples of interface layer sub systems with higher bandwidth than the mechanism under control are:

- The pressure, flow or other internal states of mechanical power control systems may have to be controlled at much higher bandwidths than the actuator itself, due to properties of the medium.
- Current control loops for electrical actuators depend on the electrical transfer function of the actuator, which may have much higher bandwidth than the mechanical transfer function.
- Switched mode power controllers for electrical actuators have typical switch frequencies of $10kHz$ to $100kHz$, and may control switching times with bandwidths in excess of $10MHz$.
- Sensor interfaces may require oversampled filters or other signal processing operating above the mechanical bandwidth.

To illustrate this we will elaborate on the linear actuator example given above, which could be implemented on a system similar to figure 3.8

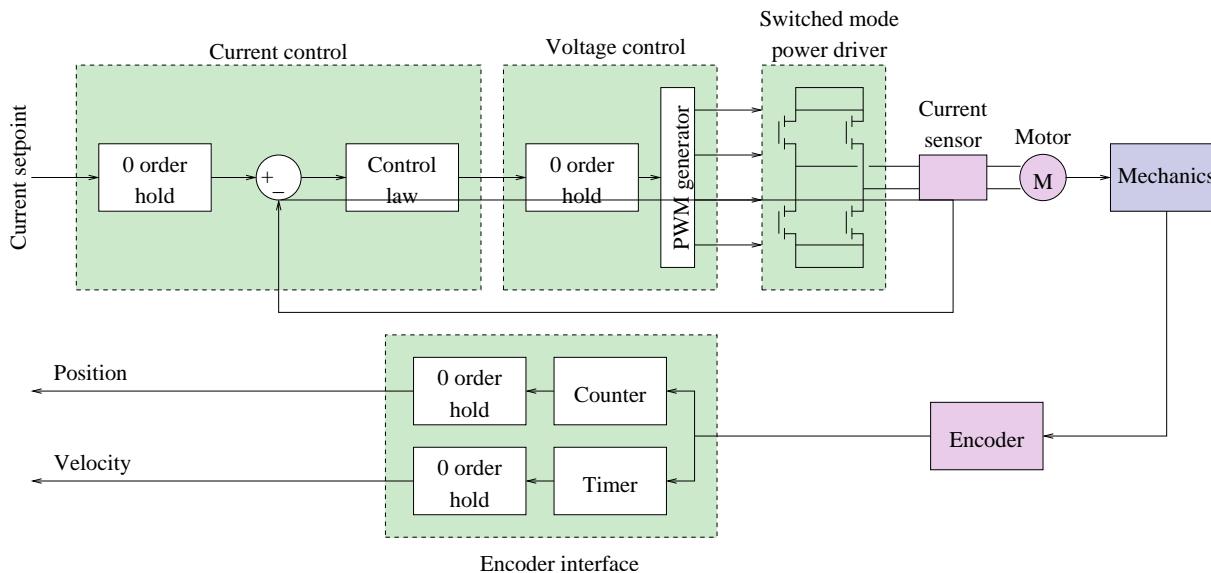


Figure 3.8: Example of interface layer sub systems

The motor is driven by a pulse width modulated (PWM) switched mode power driver, operating at a switch frequency of, say $50kHz$. If the pulse width² is to have a resolution of, say 7 bits or $42dB$, the PWM generator must have a bandwidth of $2^7 \cdot 50kHz = 6.4MHz$.

²The pulse width is equivalent to the effective mean voltage applied to the motor

The 0 order hold filter in the input of the voltage/PWM controller, allows the current control loop to operate at a lower sample rate. Let's assume the L/R time constant of the motor coils is $300\mu s$, then we might operate the current control loop at a sample rate of $10kHz$, or maybe $12.5kHz$ if we wish to synchronize to the PWM frequency. The 0 order hold filter in the input allows the control layer to update the current set point at a lower sample rate, for instance at the $3.6kHz$ that ensures the amplitude of aliasing noise to be below sensor detection. It might be prudent to choose a sample frequency that can be synchronized to the current control loop, to avoid inter modulation frequencies

The position of the actuator is monitored by an incremental encoder, that changes state, with a resolution of $2\mu m$. If the actuator can reach a velocity of $7m/s$, equivalent to full acceleration for half the length, the bandwidth of the encoder and the counter needs to be at least $3.6MHz$. If speed is to be measured, by timing the change to encoder state, an even higher bandwidth may be needed, depending on the resolution and latency of the speed measurement. The 0 order hold filters in the output of the encoder interface, means that the last sensor values can be read any time by control layer applications.

The example may be overly simplified, but it illustrates how the interface layer may contain high bandwidth sub systems and feedback loops, in order to present a useful interface for a mechanical system with a lower mechanical bandwidth.

Although we might probably rely entirely on commercial components and sub systems to implement the necessary high bandwidth systems, thus avoid involving the GEECONs in the interface loop, we find it highly advantageous to be able to involve the GEECONs for the following reasons:

- By moving some functionality from external power control components to the GEECON, we can often use simpler, smaller, lighter or cheaper power control components.
- Including the GEECON in the interface loop may simplify the design of power control components in the cases where suitable commercial components do not exist.
- Including the GEECON in the interface loop gives us a closer integration between the important control systems of the control layer and the interface layer, which may enable us to achieve better overall performance, as delays can be decreased, synchronisation can be improved, and control algorithms gain access to more states of the system under control.

If we wish to involve the GEECON in the interface loop, the I/O and signal processing systems of the GEECONs might be confronted with bandwidth requirements far beyond the mechanical bandwidth of the mechanisms to be controlled. We can only narrow these requirements down by investigating the systems to be controlled, and consider the various possible ways to control them. The best way to implement the interface loop, will then be a compromise between space, cost, performance, component availability and the capabilities of the GEECON.

It stands to reason that the best way to ensure our GEECON architecture to be general, is to apply a flexible and fast I/O technology, backed up by powerful signal processing capabilities.

Control loop bandwidth

Assuming that the interface layer hides all internal subsystem that require bandwidths in excess of the mechanical bandwidth, the control loop bandwidth only need to match the mechanical bandwidth of the actuator system. As we have seen in the discussion of the linear actuator example above, a sample rate of $3 - 4\text{kHz}$ will be sufficient for most industrial mechanics. In most cases we can allow the control loop to run much slower, as most industrial mechanics have far lower bandwidths.

At this point, it is important to repeat that the layered model is not an implementation model, and although it is an appealing abstraction to classify the low level control into nested interface- and control-loops, it might prove highly beneficial to implement these functions in a more integrated way, where the loops interact in more complex ways. In practice, the signal processing in these loops will be implemented with a mix of software, digital electronics, possibly analog electronics, and perhaps even mechanical systems, making it impossible to formulate precise performance demands for each technology in advance.

The ability to implement a position control loop with a sample frequency of at least 4kHz for each actuator system in a mechanical host module is a reasonable way to ensure that our GEECON can control a very wide range of industrial mechanics. Assuming that a mechanical host module will have at most 10 actuators, that the CPU will only have to handle a simple control law for each actuator, and allowing 25 operations for each evaluation of the control law, we will need a performance of at least 1 million operations a second. If we want to apply more complex control laws or include interface loops in the system, we may have to increase this demand dramatically, or compromise with respect to the number of actuators or the sample rate. We must also keep in mind that the GEECON have tasks beside implementing the control- and interface-layer. In other words, the requirements of the control layer points to a computer architecture that is well suited to execute repetitive algorithms with fixed rates of at least $10 \times 4\text{kHz}$, while tending other tasks as well.

To put these considerations into a practical perspective, we note that the 3 actuator DT-VGT and 7 actuator PA-10 modules chosen to evaluate our technology, will limit the execution frequency of the control loop to 2kHz and 1.5kHz respectively, due to the sample frequency of the DT-VGT feedback sensors and the latency of the native PA-10 controller. Allocating 1 million operations per second for the control layer will then give at least 166 and 95 instructions respectively to evaluate control laws for the two mechanisms.

3.5 The execution layer

The way we have defined the role of the execution layer, it plays three important roles in the data flow:

- It receives path specifications for the individual actuators from the application layers.
- It performs global kinematic compensation on the path specifications, which may be based on sensor feedback from the robot.
- It relays the compensated path specifications to the control layer.

As we have decided to distribute the execution layer via. a network, the bandwidth requirements of this arrangement is especially interesting, as it has profound influence on the choice of network technology and the system performance.

At the bottom level, the execution layer supplies the position control loops with reference paths, which are sampled with the sample frequency of the position control loop.

At the top level, the application layer supplies reference paths, with a sample rate that might be substantially lower than the mechanical bandwidth, and the position control loop. This is due to the fact that application layer components usually operates with very conservative limits to actuator motion compared to the physical limits that define the worst case mechanical bandwidth.

For generic motion planners, it is common practice to represent robot positions as a joint vector $\mathbf{q} = \{q_1 \dots q_n\}^T$, where q_i is the position of the i'th joint and n is the number of joints in the robot. If the robot is to use a tool, it is convenient to include the tool parameters in the joint vector as if the tool parameters were simply additional joints. Depending on the application, the joint position or tool parameter can be accompanied by 1 or 2 derivatives \dot{q}_i and \ddot{q}_i , and possibly other useful parameters, such as an estimate of joint force/torque τ_i . It is also common practice that robot paths are defined as a stream of joint vectors given at equidistant time intervals.

We expect the sample rate of the *joint stream* from the application layers to be the lowest possible that is able to give a satisfactory representation of the actuator paths, which means that one of the main objectives for the execution layer is to act as a sample rate converter. We see no reason to perform kinematic compensation after the sample rate conversion, and assume that the compensation will run synchronous with the joint stream. This allow us to allocate the sample rate conversion to the GEECON, as indicated in figure 3.10.

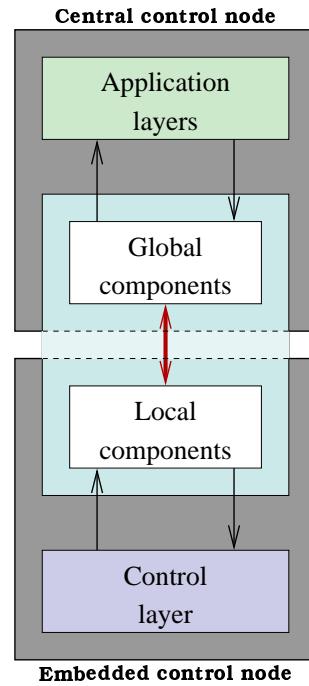


Figure 3.9: Distributed layer

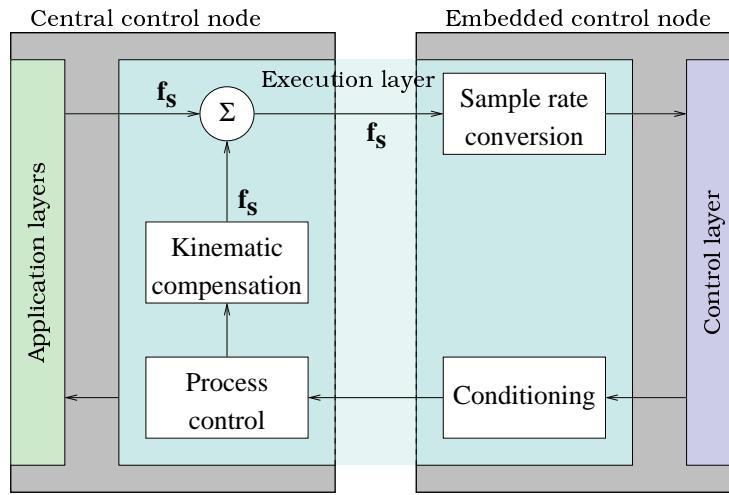


Figure 3.10: Model of execution layer

By maintaining the same, lowest possible, sample rate f_s in the local components of the execution layer, we minimize the amount of data to be transmitted via the network.

The amount of feedback data needed for central execution layer and application layer components depend on the application, and it may not be practical to use the same sample rate for these as for the joint stream. If it is possible for the GEECON to pre condition these data in a way that reduces the network load, we might wish to include such conditioning components in GEECONs.

Demands for the joint stream

Assuming that the numerical resolution of the joint stream is much higher than the sensor resolution of the mechanisms, the resolution of a path specification is equivalent to its sample rate.

The sample rate conversion will reconstruct low resolution path descriptions into high resolution ditto, to be used as reference path for the control loop. This reconstruction is equivalent to the reconstruction performed by a D/A converter [Ifeachor93].

Practical motion planners operate with limitations to both velocity and acceleration. Partly to avoid exceeding the physical limits of the robot under control, and partly to oblige process parameters. These limits will constrain the frequency content of the motion planner output to:

$$Q(f) \leq \frac{|\dot{q}|_{max}}{2\pi f} \quad \text{and} \quad Q(f) \leq \frac{|\ddot{q}|_{max}}{(2\pi f)^2} \quad (3.4)$$

As illustrated in figure 3.11, the bandwidth of a joint stream can then be defined as the crossing point between the lowest limit and the acceptable joint position tolerance Δ_x .

If the sample rate converter were able to perform an ideal reconstruction, the sample frequency of the joint stream could be chosen as twice the bandwidth, but as this is practically impossible, we are forced to require a higher sample rate.

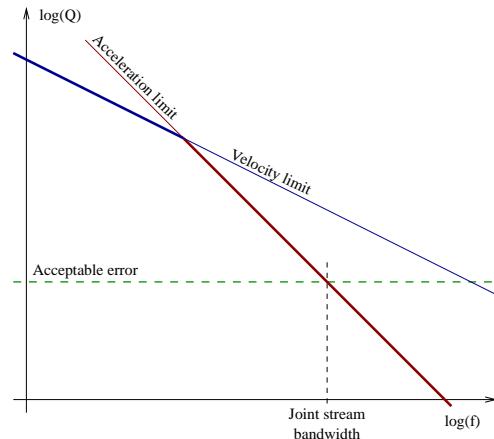


Figure 3.11: Joint stream bandwidth

As the sample rate converter is likely to be part of a feedback loop, it is under much tighter constraints with respect to signal delay than if we were designing an open loop system, such as audio playback. Any signal delay will cause a phase shift, that will decrease the stability of the closed loop. Delays are usually caused by interpolation, FIR filters, and similar algorithms needing a number of samples to operate, and can be expressed in terms of sample periods. If we want a limited phase shift, the Nyquist frequency $f_s/2$, expressing the highest theoretical bandwidth, is replaced by the following:

$$f_\phi = \frac{f_s \cdot \phi}{2\pi \cdot N} \quad (3.5)$$

Where f_ϕ is the highest frequency with a phase shift below ϕ , f_s is the sample frequency, and N is the number of samples the signal is delayed. Evidently the joint stream have to be oversampled by a factor of $N \frac{\pi}{\phi}$ to maintain the same useful bandwidth as a delay free system. This will increase the network load as well as the computational load of higher layers, and we are urged to choose a reconstruction method with minimal delay.

Extrapolation will reconstruct the path, with less than one sample period of delay, and is an excellent candidate reconstruction method. Zero and first order extrapolation, as illustrated in figure 3.12 will reconstruct the path with an error of up to $|\dot{q}|_{max} \cdot t_s$ and $|\ddot{q}|_{max} \cdot t_s^2$ respectively. The error of zero order extrapolation can conveniently be expressed as a delay of half a sample period combined with a position error of up to $\frac{1}{2}|\dot{q}|_{max} \cdot t_s$

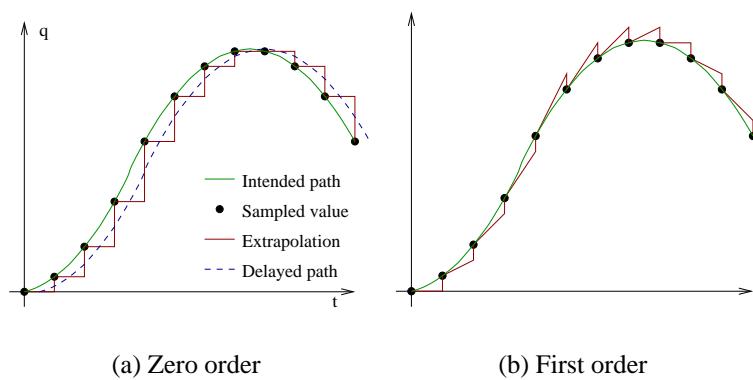


Figure 3.12: Extrapolation

The extrapolation errors effectively form a sawtooth signal superimposed on the intended path. The frequency spectrum of a sawtooth with peak-peak amplitude A is:

$$\frac{A}{\pi} \left(\frac{\sin w}{1} - \frac{\sin 2w}{2} + \frac{\sin 3w}{3} - \dots \right), w = 2\pi f_s t$$

Where it is evident that the amplitude of the harmonics of the sample frequency only subsides linearly with frequency.

The continuous output of an interpolation reduces the noise at the cost of signal delay. Using a first order interpolation, as shown in figure 3.13 introduces a signal delay of one sample period, and maximum position error of $\frac{1}{8}|\ddot{q}|_{max} \cdot t_s^2$

The worst case error signal is a periodic parabola piece, which is the integral of a sawtooth. In this case, the harmonics of the sample frequency subsides with the square of the frequency, as the frequency spectrum of a worst case error signal with peak-peak amplitude A is:

$$\frac{4A}{\pi^2} \left(\frac{\sin w}{1^2} - \frac{\sin 2w}{2^2} + \frac{\sin 3w}{3^2} - \dots \right), w = 2\pi f_s t$$

Interpolation of higher orders will reduce the error amplitude even further, at the cost of further delay, and might be worth considering in systems where delay is not critical.

Extrapolation and interpolation as discussed above are simple and straight forward to understand and implement. Other methods exist, and the methods can be combined in various ways, but

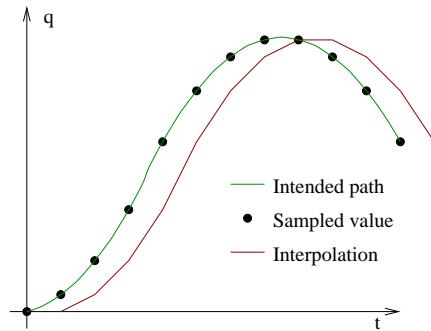


Figure 3.13: Interpolation

no matter what reconstruction method is used, the three main criterions in choosing the sample frequency remains:

- The sample frequency must be sufficiently high to represent the intended actuator path with the desired accuracy. This criterion can be established at the central control node, as it has nothing to do with reconstruction, but depends only on the velocity and acceleration limits in the motion planner as well as the demands for process precision.

- The sample frequency must be sufficiently high to avoid instability due to signal delays in the reconstruction. To establish this criterion we require information about acceptable phase shift from the central control node, as well as information about reconstruction delay from the GEECONs.

- The sample frequency must be sufficiently high to suppress the reconstruction error to a level that is acceptable to the lower layers of the robot controller. Although a certain level of position error might be acceptable to the process, the accompanying oscillations at the sample frequency and its harmonic frequencies might have adverse effects on the lower level control systems, power consumption and mechanical durability. To establish this criterion we require information about the acceleration and velocity limits from the central control node, while the GEECONs must submit information about the reconstruction method, and acceptable levels of reconstruction errors.

As an example, consider a system with a linear actuator, where the motion planner has joint limits of: $|\dot{q}|_{max} = 3\frac{m}{s}$, $|\ddot{q}|_{max} = 30\frac{m}{s^2}$. The process allows a tolerance of $1mm$, but the actuator should not be submitted to harmonic frequencies with an amplitude above $10\mu m$. The control loop has a transfer function which can be enveloped by a 1. order low pass filter with a corner frequency of $f_c = 10Hz$.

The joint stream bandwidth is then limited by the acceleration to just below $28Hz$, which implies a sample frequency of $f_s \geq 56Hz$. Table 3.1 shows the magnitude of the reconstruction error, worst case actuator disturbance due to the error, and resulting minimum sample frequency, signal delay, and corresponding phase delay at the joint stream bandwidth, for different reconstruction methods.

Reconstruction	p-p error	harmonics on actuator	sample freq.	delay	phase
0. order extrap.	$\frac{ \dot{q} _{max}}{f_s}$	$\frac{ \dot{q} _{max} \cdot f_c}{\pi f_s^2} \sum_{N=0}^{\infty} \frac{1}{(1+N)^2}$	1.3kHz	$\frac{t_s}{2}$	4°
1. order extrap.	$\frac{ \ddot{q} _{max}}{f_s^2}$	$\frac{ \ddot{q} _{max} \cdot f_c}{\pi f_s^3} \sum_{N=0}^{\infty} \frac{1}{(1+N)^2}$	151Hz	0	0
1. order interp.	$\frac{ \ddot{q} _{max}}{8f_s^2}$	$\frac{ \ddot{q} _{max} \cdot f_c}{2\pi^2 f_s^3} \sum_{N=0}^{\infty} \frac{1}{(1+N)^3}$	115Hz	t_s	87°

$$\sum_{N=0}^{\infty} \frac{1}{(1+N)^2} \simeq 1.65 \quad \sum_{N=0}^{\infty} \frac{1}{(1+N)^3} \simeq 1.2$$

Table 3.1: Reconstruction method vs. sample frequency example.

For most practical robot applications, the above example represents extreme limits for the velocity and acceleration limits combined with the process accuracy, and the limit to harmonics. Only to be able to envelop the transfer function of the control loop in a 1. order low pass filter, is also pessimistic from a practical point of view. The example therefore serves to demonstrate that sample rates of a few 100 Hz will be sufficient for joint stream to control practical aggregated robots for industrial processes, if we use 1. order extra- or interpolation to reconstruct the reference paths for the control loop.

Network bandwidth

Assuming we need sample rates of up to 200Hz for the joint streams, that joint positions can be adequately represented as 32 bit numbers, and that practical robot systems will contain no more than 25 individual actuators, the bandwidth requirements to transmit joint streams will be below 200,000 bits per second. Including e.g. 2 derivatives and a force/torque estimate in the joint stream will then drive the bandwidth requirement as high as approximately 1 megabit per second.

We see no reason for the amount of motion feedback to exceed the motion specification, and assume that the bandwidth requirements for motion feedback are less than or equal to the demands for the joint stream.

We can not assume much about the amount of feedback apart from actuator feedback. While most applications will require modest process feedback, equivalent to a few extra actuators, we can easily imagine applications requiring vast amounts of feedback from external sensors such as cameras, scanners etc. The only way to ensure generality is to supply infinite communication bandwidth, but in practice we must compromise between the quest for bandwidth and practical concerns such as availability, robustness, cost etc.

Network latency

As the joint streams may participate in various forms of closed loops, we must also consider the network latency, as transmission delays will cause phase shifts that affect the stability of such loops.

We can not foresee which latency demands future applications may place on the network, but we suggest that the latency is kept as low as possible, and at least an order of magnitude below the sample period of the joint stream. As we have estimated the highest relevant sample rate to 200Hz, this translates to a worst case latency below $500\mu s$.

The latency from central controller to different GEECONs may be different in e.g. network technologies based on point-point connections. Such latency differences may affect the synchronisation of actuator movement, which must be considered. The latency difference can never surpass the worst case latency of $500\mu s$. With accelerations below $30m/s^2$, the worst case position error due to the synchronisation error will be less than $4\mu m$, which we find quite acceptable.

The same estimate holds for latency variation, or jitter. The Jitter will never surpass the worst case latency, and position errors due to jitter will also be below $4\mu m$, if acceleration is below $30m/s^2$.

3.6 Conclusion

In this chapter, we have proposed a distributed architecture, that allows us to take advantage of the modular nature of aggregated robot system, by integrating GEECONs (generic embedded control nodes) in each mechanical module of the system, and coordinating them from a central control node that implements the global functions of a controller.

We have used the layered reference model as basis for a discussion about the roles and interfaces between the mechanical modules, their native control technology, and the GEECONs. We proceeded to consider the roles and interfaces between the GEECONs and the central control node in order to establish their overall relationship.

We have discussed the information flow of a robot control system, in order to elaborate on the speed, bandwidth and latency requirements of the basic components in our proposed architecture, in relation to likely and practical use, dwelling to explore the relationship between bandwidth or sample rate, and physical properties, demands and constraints of various levels of a robot control system.

Using a number of abstract examples to encompass the likely scope for a modular industrial controller, we have quantified some important requirements, while we have found others to depend too much on the practical applications to be quantified in advance.

Our major conclusions with respect to the components of a distributed robot controller are:

- The generality of a GEECON can only be assured by invoking an I/O interface capable of interfacing to all manner of official and de facto signal standards. We should therefore aim for as flexible an I/O system as practically possible.
- If we rely on external components and subsystems to implement all the interface layer components, an I/O sample rate of 4kHz should be enough to ensure that we can control most practical industrial mechanics.
- As we can benefit from integrating the interface loop with the GEECON, and as many interface layer components may require bandwidths well into the MHz range, we should aim for as high an I/O bandwidth as practically convenient.
- Conducting position control with sample rates of up to 4kHz should ensure that we can control most practical industrial mechanics. A computational power of 1 million operations per second will therefore be sufficient to control the actuators of a single module, using simple control laws.
- In addition to evaluating control laws, computational power must be allocated to communication, execution layer tasks, and system control, and possibly also to implementing interface layer functions. In order to allow all these functions, and to allow use of more demanding control laws, the performance of the computer platform used for GEECONs should be well in excess of the power needed for simple control.
- The GEECON must perform a sample rate conversion, to reconstruct high resolution reference paths for the position control, from minimal resolution path specifications delivered via network from the central control node.
- Sample rates up to 200Hz should ensure sufficient path resolution for most practical industrial mechanisms. A network performance of up to 2Mbps should therefore cover the requirements of motion specification and feedback for most conceivable aggregated robots, if we exclude the demands for external sensor data.
- Network latencies below $500\mu s$ should ensure sufficiently low phase shifts, sufficient synchronisation and sufficient suppression of jitter, to convey path specifications and feedback within the requirements of most conceivable aggregated robots.
- In order to allow external sensor data to be communicated through the *backbone* network of our architecture, the network performance should be as high as practically convenient.

Chapter 4

Embedded platform

Abstract

Description of the selected architecture and technology for the generic embedded control nodes (GEECONs).

The introduction of generic embedded control nodes (GEECONs), that implement all the local functions of a robot controller, for individual mechanical modules of the robot, is the basis for our proposed controller architecture.

We have already established the GEECONs primary roles as interface, motion controller, sample rate converter as well as system controller, database and communication node. These functions require a computer platform along with an I/O and network interface and some suitable software. We have — not surprisingly — established that the generality of our controller is linked to the flexibility, bandwidth and computing power at our disposal, and have developed a rough estimate for the requirements of typical industrial mechanics.

In order to develop a successful concept for the GEECONs we also have to consider a range of other issues related to the integration with host modules and the operating environment, such as size, shape, power supply, vibration, mechanical- and electromagnetic environment.

4.1 Physical demands

This project represent the initial stage of a development, and is primarily aimed toward development and research in modular robotics. Our development and research do however involve industrial partners, and we do wish to be able to conduct tests and experiments in realistic industrial environments. As our development progresses, it is also a specific goal to evaluate and use our technology in prototype robotic production equipment. This gives us the following overall constraints:

- We must be able to physically integrate the prototype GEECONs of this project with the DT-VGT and PA-10 mechanisms, that are used as evaluation modules for this project,
- Our prototype GEECONs must be based on technology that can realistically be integrated with other mechanical modules used or considered by our industrial partners.
- The prototypes of this project need not comply with requirements for sustained use in a heavy industrial production environment, as they will only be used for experiments and demonstrations.
- Our prototypes must be based on technology that can realistically be implemented in compliance with demands for sustained use in heavy industrial production equipment.

This means that our primary concern is to identify a computer platform technology that is powerful, yet small and flexible enough, to satisfy our major demands.

As discussed in chapter 2, previous examples of generic controllers, was oriented toward robot cells, rather than robot modules, and it could be assumed that they could be placed external to the robots using conventional electronics enclosures based on cabinets or racks. Moving the computer into the body of the robot presents us with more strict demands for size and shape.

The only way to ensure complete generality is to demand the GEECONs to be infinitely small. During this project we will assume that the demands posed by our two example mechanisms are representative for typical industrial systems.

DT-VGT

The demands for integration with the DT-VGT prototype are instructive, as the DT-VGT allows very little room for integration compared to systems previously encountered.

The DT-VGT offers no possibility to draw on existing support systems such as power supply, enclosure etc. as it has no native controller. Rather it forces us to implement a complete controller node, including power control components, power supply and other relevant support systems, in a quite limited space.

Due to the geometry of the DT-VGT, and the intention to use it to enter narrow spaces, the only place to put the controller node is inside the tube that forms the static part of the mechanism, allowing us a cylindrical space with diameter: 129mm and length: 330mm to implement power supply, power control, I/O interfaces and control computer. In addition to the control node, all wires, hoses etc. associated with the DT-VGT must pass through the same cylinder as it provides necessary mechanical protection.

It will be most convenient to mount the electronics in a rectangular enclosure inside the cylinder, partly because rectangular geometry is preferable for electronics, partly because a rectangular box will allow cables and hoses to pass between the box and cylinder walls. Using a dedicated enclosure for the electronics will also simplify proper environmental sealing when the system

must be implemented for continuous use in industrial environments. Reserving space for external connectors etc. we can use a box with external dimensions of up to : $91 \times 91 \times 250\text{mm}$ giving us an internal space of approximately $88 \times 88 \times 247\text{mm}$ to roam.

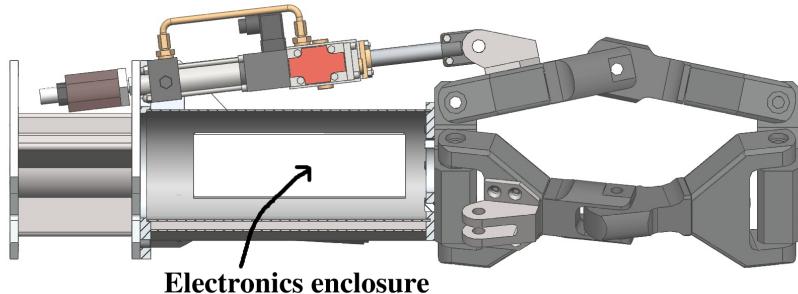


Figure 4.1: Artists impression of enclosure inside the VGT structure

PA-10

As the PA-10 has a native controller with ample surplus space, any controller that can be fitted into the DT-VGT can also be fitted into the PA-10 controller.

4.2 Computer platform

In reality, no standardised computer platform is useful for our purpose. Our demands for compactness immediately disqualifies popular modular computer architectures based on backplanes, such as: VME [VITA], ISA, PCI [PCISIG], and Compact-PCI [PICMG]. The more compact ISA/PCI derivative PC-104/PC-104+ [PC/104] is not compact enough, and can also be discounted.

Leaving the realm of standardised industrial architectures, we can either choose from a multitude of commercially available non standardised miniature computers, or we can design our own if no suitable commercial product is found. In either case, the selection and design criterions are similar, with primary focus on the CPU technology and associated peripheral and support systems.

The market for CPU technology is exceedingly complex, as the number of combinations of technology, architecture, word size, speed, and I/O bandwidth is practically infinite, and thousands of different products exist, covering most useful combinations of basic parameters.

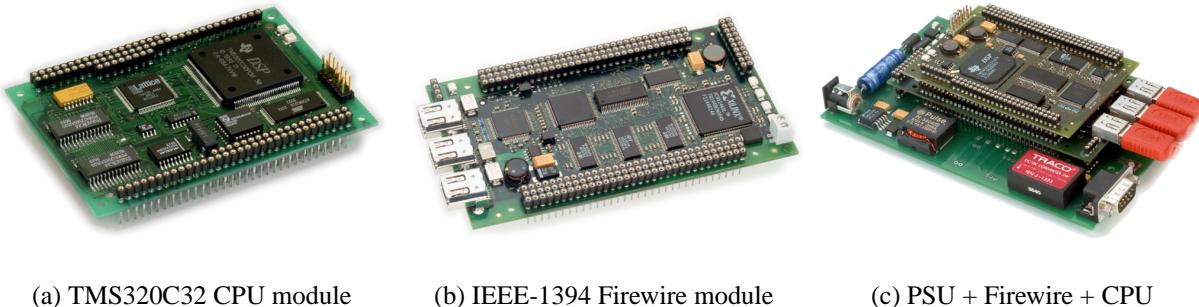
There is no obvious way to decide the optimal CPU technology for our GEECONs, but our wish for a simple and small system with high I/O bandwidth and good performance for highly repetitive algorithms point towards digital signal processors, or possibly RISC based micro controllers.

We have evaluated various commercial products based on DSP's or micro controllers in order to establish if a suitable technology exists, and have decided to build our prototypes over the Microline CPU modules sold by the German company Orsys GmbH.

Microline platform

Orsys GmbH has specialized in producing evaluation systems for various Texas Instruments Digital signal processors and accompanying peripherals. Orsys product range consist of a small number of CPU modules, containing a Texas Instruments DSP, along with memory, a RS-232 interface, and a bus interface to their own *Microline Bus*, which is a simple extension of the DSP's peripheral bus. The Microline Bus connectors are mounted vertically along the board edges, so connecting modules are stacked, in the same manner used by PC-104 [PC/104]

In addition to the CPU modules, Microline also manufactures a small selection of I/O modules, among which is an IEEE-1394 Firewire interface, which is described in chapter 6.



(a) TMS320C32 CPU module (b) IEEE-1394 Firewire module (c) PSU + Firewire + CPU

Figure 4.2: Various Microline products

With a board size of $98 \times 66\text{mm}$, the Orsys CPU modules can easily be accommodated by the DT-VGT. Each Orsys board occupies 11mm of height when they are stacked, giving us room for a stack of up to 6 boards.

The main features of the Microline range of CPU's are:

- 32-bit 20MHz Microline peripheral bus
- 2 - 16 DMA channels
- 40-900 MFLOPS
- 20-1336 MIPS
- 128K - 2MB of RAM.
- 128K - 512K of boot FLASH.
- Low Power consumption $< 10\text{W}$
- Acknowledged by Texas Instruments
- Supported by TI's SW development tools
- Programmable in C as well as assembler

The entire range of Microline CPU modules surpasses the overall performance considerations established in the last chapter with a wide margin, and apart from a few correctable drawbacks, the technology is quite suitable for our purpose. The major drawbacks are:

- The connectors used for the Microline bus are not suited for industrial environments, as they are prone to long term failures when submitted to vibrations.
- The Microline bus has a poor EMC design with excessive loop areas for sensitive signals
- The selection of I/O boards is virtually nil
- The available power supply boards are too large

The connector quality does not concern us during this project as the prototype controllers are only subject to industrial conditions for short intervals during testing. When an industrial version is to be implemented, the connectors can be exchanged, or the electronics can be re-engineered, depending on the amount of controllers to be manufactured.

The poor EMC design does not affect operation under low noise conditions, but increases the susceptibility to magnetic and electromagnetic coupled noise. The problem can be remedied with proper shielding, by manually adding external ground connections between connecting boards, or by redesigning the electronics.

The poor selection of I/O boards forces us to implement an interface to a useful and popular I/O standard, or to implement our own I/O boards. We will discuss this issue in chapter 5.

The size of the Orsys power supply boards is a minor problem, as a suitable power supply board can easily be implemented.

4.3 Architecture

The three major components in our basic architecture are the CPU module, the backbone network interface, and the I/O interface. Having decided to go with the Microline CPU modules, we have investigated several possible ways to implement the remaining network and I/O components in a compact, efficient and flexible way. A lot of our resources have been allocated to this development, which have involved several separate iterations over both network and I/O, before bringing them together in the architecture outlined in figure. 4.3.

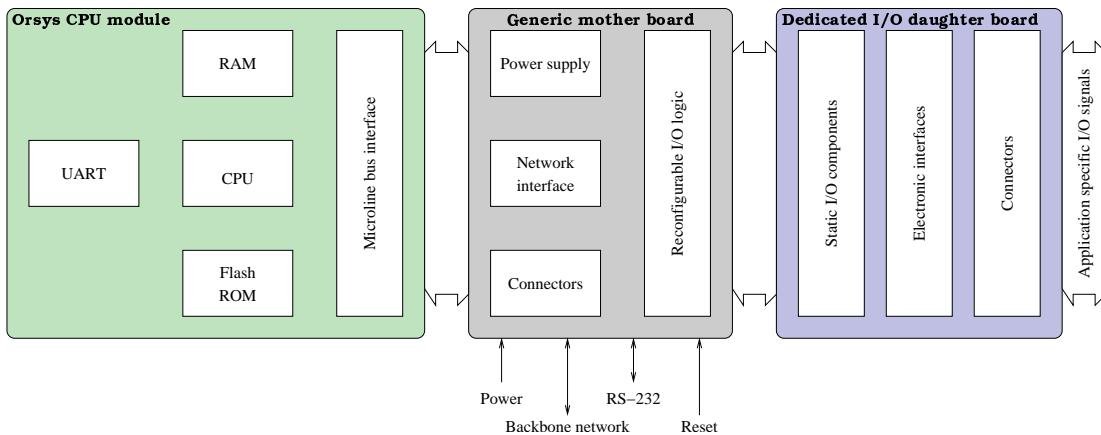


Figure 4.3: Architecture of GEECON

Our considerations regarding the details of the backbone network and I/O system are described in separate chapters. Here we will only describe how these systems have been integrated into a flexible, yet powerful and compact unit, involving a total of only 3 circuit boards:

- Mother board
- Microline/Orsys CPU module
- I/O daughter board

Mother board

All components which are independent of the host mechanism have been assembled on a single circuit board that act as host for a CPU module and an application specific I/O daughter board. This *mother board* implements the following functions:

- Power supply
- Network interface for the backbone network
- Reconfigurable I/O logic
- Connectors for power, network, RS-232, and Reset.

The mother board is the same size as the CPU module, and is configured with connectors for the Microline bus on one side, and a set of connectors for the daughter board on the other side, effectively forming the center of a *sandwich* between a CPU module and an I/O daughter-board. The documentation for the mother board can be found in [Kyrping-A].

Power supply

As the GEECONs are to be integrated with many different technologies, the characteristics of the available power supply may vary from application to application. In industrial applications we will typically encounter voltage supply's of: 400VAC, 230VAC, 200VAC, 110VAC, 48VDC, 24VDC, or 12VDC. In an industrial environment, any supply voltage used for power distribution is prone to variations and noise.

The 3 boards of our GEECON requires 5V and $\pm 12V$ with minimal variation and noise, which is provided by a set of integrated DC-DC converters on the motherboard. One supplies +5V, one supplies $\pm 12V$, and one is used to generate a supply voltage for external purposes, typically 24V. The last can be omitted if all necessary voltages are readily available from other sources. All three DC-DC converters are of the same type, and are all rated for 10W of output.

We have decided to use 48V DC-DC converters with an input range of 36...75V for the following reasons:

- In cases where the supply voltage is to be distributed and used for other purposes than supply for GEECONs, it is more efficiently distributed at high voltages.
- 48V is just low enough to avoid hazardous voltage restrictions.
- 48V is used by the power amplifiers we have developed for the DT-VGT. Adopting the same voltage as supply for the GEECONs saves critical space in the DT-VGT application.

In applications where 36...72V is not readily available, the DC-DC converters of the motherboard must either be exchanged, or an appropriate voltage must be generated from the available power supply.

Network interface

The ARC-net interface used as backbone network is implemented using a dedicated ARC-net controller IC [COM20022] which is directly compatible to the Microline bus. The controller IC and accompanying transceiver is placed on the mother board, and is directly connected to the Microline bus.

Our considerations and experience concerning the backbone network are described in chapter 6.

Reconfigurable I/O logic

As described in chapter 5, the I/O system that connects the GEECON to the host mechanical host module is based on reconfigurable logic, implemented with an Field Programmable Gate Array (FPGA). The FPGA implements the I/O logic of the application in question, while the necessary signal conditioning etc. is performed with the dedicated circuitry of the I/O daughter-board.

The FPGA is placed on the motherboard, which connect relevant signals of the Microline bus to its I/O pins, enabling the FPGA to perform as a multi purpose peripheral to the CPU. A few FPGA I/O pins are allocated for status and debugging purposes, but the bulk of FPGA I/O pins are connected to the I/O daughter-board through two vertical connectors.

I/O Daughter board

In order to connect the FPGA I/O signals to the external world, a certain amount of signal conditioning, signal conversion, filtering, buffering etc. is necessary. This depend entirely on the application, which is why we have relegated these functions to a separate board that can be specifically designed for a given application without compromising the generality of the overall architecture.

The Daughter board developed for the DT-VGT is equipped with CAN-BUS controllers and transceivers, an analog to digital converter with appropriate analog signal conditioning, and a number of buffers for digital input and output signals. The DT-VGT daughter board is described in more detail in section 7.4.

It has not been necessary to develop a daughter board for the PA-10 application, as we are using a prototype Microline/ARC-Net board to communicate with the native PA-10 controller.

The CPU module

The Microline CPU modules are downward compatible, which makes it possible for us to upgrade the CPU module if the need arises. Although we chose the comparably modest 40MHz/40MFLOP Microline C32 board [Orsys-A] at the beginning of the project, we have not yet had reason to upgrade to a faster type. The C32 board is build over a TMS320C32 DSP, with 2MB 1 wait-state external RAM, and 512KB external flash ROM for program storage etc. The CPU module is also equipped with an RS-232 interface (UART) used for programming and program interaction. The Microline bus is a simple derivation of the TMS320 peripheral bus, and the Microline interface consists of a few line transceivers and some combinational logic. The timing of the bus interface is poorly documented, and we had to reverse engineer the necessary timing specifications from the component data sheets in order to design and implement peripheral components. [Sørensen-B] [Sørensen-C]

4.4 Experience

The platform development for the GEECONs have been both successful and encouraging. The FPGA based I/O system have allowed us to implement a system which is compact, efficient and provides an elegant software interface, allowing very simple and efficient low level software.

Although we have concentrated on integration with the DT-VGT platform, experiments have been conducted with other applications for our system as well as its individual components. We have:

- Supervised the successful integration of a GEECON with the native controller of a PA-10 robot — see chapter 8.
- Composed the DT-VGT and the PA-10 into an aggregated robot, controlled by two GEECONs, and used it to demonstrate an industrial process (spray painting a wheel barrow) — see chapter 9.
- Demonstrated the flexibility of the GEECON, by offering the technology to other projects. Most noteworthy is the use of our GEECON as interface for a prototype ultrasonic scanner, where we have demonstrated the ability to perform I/O operations at a rate up to 60 Mhz.

At this time, we have tested and verified the network components of the GEECONs, as well as the central controller, but we have not yet had the opportunity to integrate them into a truly distributed system.

The compact and efficient technology that have been developed for the GEECONs, demonstrate the strength of an interdisciplinary overview and approach to computer systems design.

4.5 Conclusion

The generic embedded control node (GEECON) technology developed in this project complies with or exceeds all our demands for size and performance, and we have demonstrated that it can be used to control both the DT-VGT and the PA-10 robot used as demonstrators in this project. We are impressed by the flexibility of our generic I/O system, and are confident that it allows us to interface the GEECON to any industrial mechanism we may encounter in the future.

We are aware that the current implementation does not meet industrial demands for mechanical robustness, but the technology itself does not prevent industrial implementations, and can easily be upgraded for industrial use.

Chapter 5

Generic I/O

Abstract

The quest for a generic, yet compact I/O system has led us abandon the reliance on specialized peripheral IC's and modules, in favor of reconfigurable logic, which can be configured for almost any conceivable I/O function. An additional benefit of this approach is that any I/O function can easily be ported to other computer platforms

Computer designers have always faced the dilemma between modularity and integration. The two design strategies are almost mirror images with respect to benefits and drawbacks, and the fate of many computer platforms have been decided by the balance between the two.

Table 5.1 indicates the main properties of the modular versus. integrated design paradigms.

The modular paradigm is excellent when dealing with complex systems, where flexibility with respect to expansion and upgrades are important. The weak point of the modular paradigm is the underlying infrastructure, which will limit the extent of expansion, add complexity and overhead, as well as requiring additional space for electronic and mechanical interfaces. The need for mechanical interfaces, such as connectors, will also add to the vulnerability of the system.

The integrated paradigm is good when the system functionality is known in advance, and not likely to change over time. As there is no need for internal module interfaces, an integrated system can be made much smaller and cheaper than a modular one.

Property	Modular system	Integrated system
Design	Architecture, then module by module	Total system
Expansion	Add new modules	difficult
Upgrade	Change modules	difficult
Cost	High	Low
Space consumption	High	Low

Table 5.1: Comparison of modular versus. integrated systems

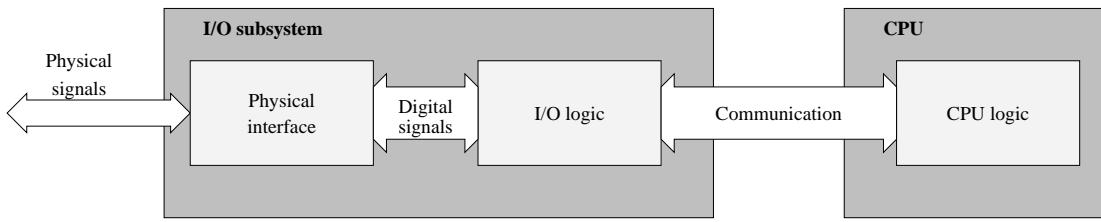


Figure 5.1: The fundamental components of I/O

5.1 The fundamentals of I/O

The purpose of an I/O subsystem is to facilitate interaction between the CPU and the external world. In order to do this, the I/O system needs two fundamental components.

1. A physical interface, to translate between physical electrical signal values — voltage, current, resistance etc. and the relevant digital representation of these signals.
2. I/O logic, to store, transform and control the digital representation of I/O information, and present/sample it to/from the physical interface.

In addition to the two fundamental I/O functions, we must also provide some channel of communication between the CPU logic and the I/O logic.

Practical I/O components have a very tight integration between the I/O logic and the physical interface, as indicated in the D/A converter example in figure 5.2. In most cases the I/O logic comes fully integrated with the essential parts of the physical layer, in a single integrated circuit. Such I/O IC's will typically represent physical signals in some *canonical* form — usually a voltage — that must often be amplified, converted or conditioned in some other way before it is suitable for connection to the outside world. We will refer to this part of the physical interface, as *canonical*.

If the D/A converter in figure 5.2 is to drive a load, e.g. a motor, its output must be appropriately conditioned, using e.g. a current amplifying transistor. We will refer to this part of the physical interface as *signal conditioning*.

In addition to signal conditioning, practical I/O systems need an electromechanical interface, in the form of suitable connectors etc. to allow connection of the electrical signals to or from the I/O system. We will refer to this part of the physical interface as *mechanical*.

While the I/O logic of figure 5.2 is rather simple compared to the physical interface, most I/O subsystems have more complex I/O logic. Partly because they perform more complex I/O operations, and partly because complexity may be successfully transferred from the physical interface to the I/O logic, to enhance performance and lower cost.

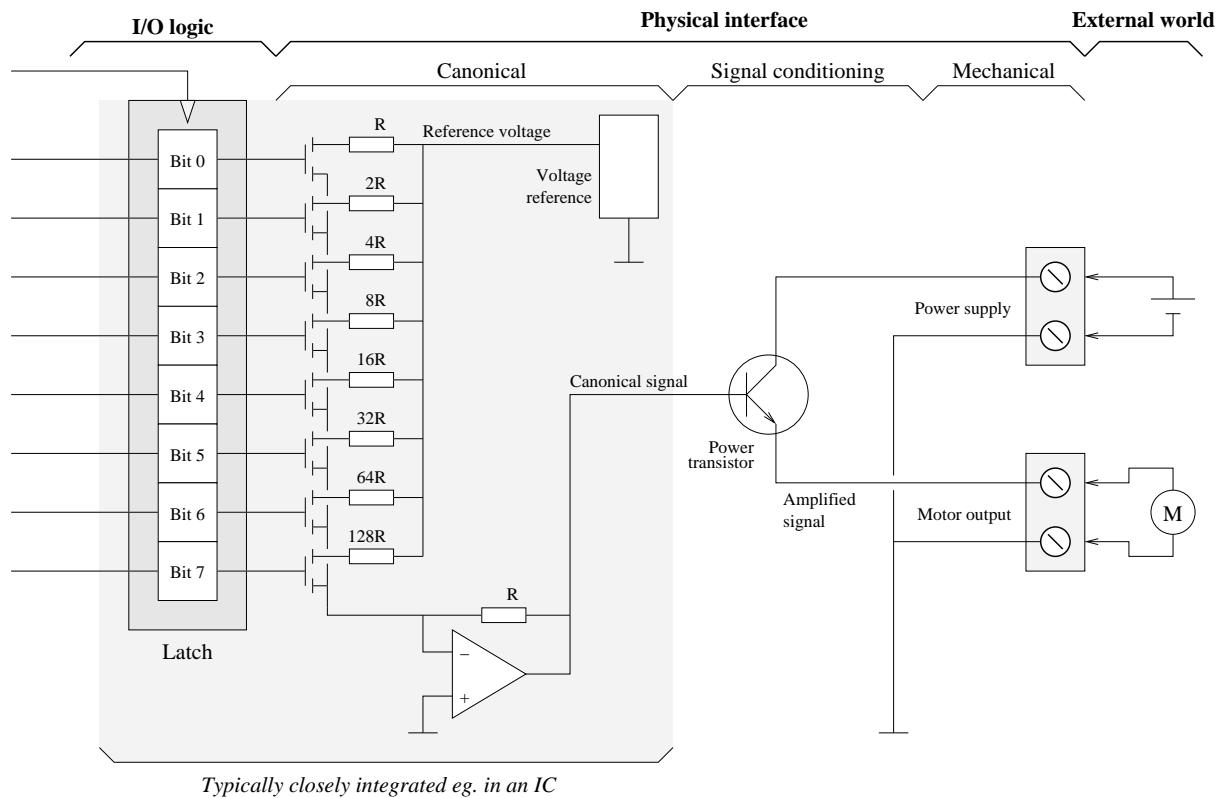


Figure 5.2: D/A converter example

The CPU to I/O link

Practical CPU's communicate with external resources — memory and I/O subsystems — through a common communication bus. In order to exchange information with a CPU, I/O systems must have a bus interface that is compatible with the CPU with respect to logic levels, protocol and timing.

The level of modularity in a traditional I/O system is primarily determined by the way the communication bus between CPU and I/O components is designed.

In non modular systems, the I/O components are connected to the bus on PCB level, with no means of exchanging or reconfiguring the I/O components. A typical example of this is a computer main-board, with integrated serial ports, printer port, network interface etc. In this case the I/O components associated with on-board I/O are connected to the CPU in a fixed way. This approach to I/O is both cost- and space effective, but requires the I/O requirements of the system to be fixed and known in advance, as it is completely inflexible.

In traditional modular I/O systems, the modularity is achieved by integrating a suitable connector scheme with the bus, letting a connector be the interface point between the bus and the I/O subsystem. This approach gives a very flexible architecture, but requires design effort, components

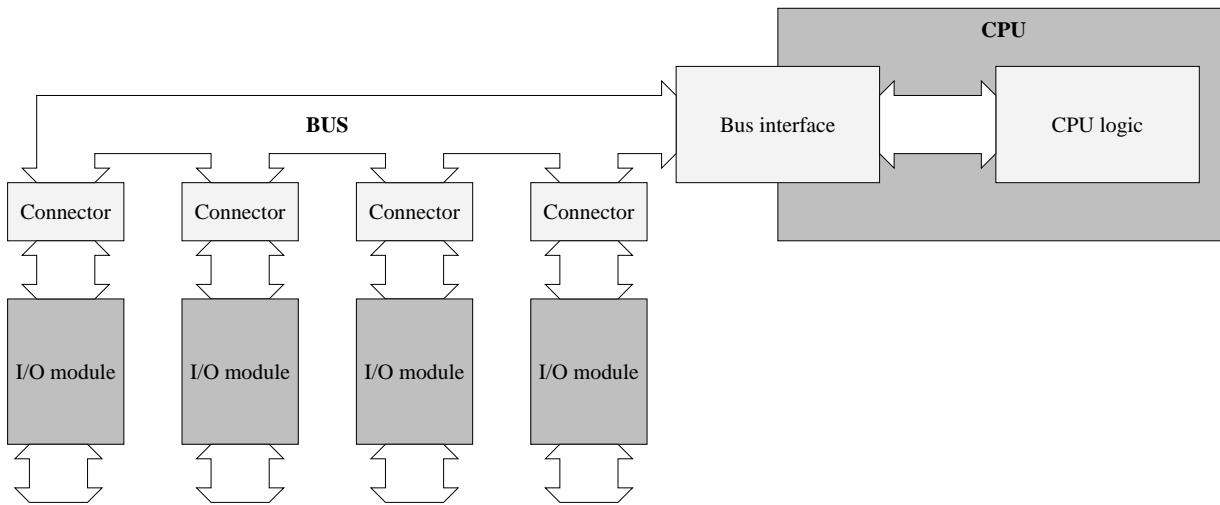


Figure 5.3: Typical modular I/O system

and space for the bus interface, adding expense and space requirements to the system.

The bandwidth and electronic reliability of such a system is largely a function of the number of connector slots for I/O modules, combined with technology used for bus drivers and bus transmission lines, including connectors. The mechanical reliability is primarily a function of the connector quality and the quality of the mechanical fastening of the I/O modules.

Traditional modular I/O systems

The bus-level modularity have been dominant in computer architecture for decades, and a large number of official as well as de facto standards for bus interfaces have evolved over the years. Among the most popular we find:

ISA is a de-facto standard for I/O as well as memory modules for personal computers. It is closely related to the memory and I/O bus specifications of the Intel 8086 processor, and can be implemented using only a handful of components in addition to the 8086. There are few demands on size and shapes of modules, as long as they can support the bus connector.

ISA bus is intended for home use and office use, and is mechanically unsuited for industrial use.

PCI Is an official bus standard for personal computers [PCISIG]. As it's bandwidth is superior to ISA, it has largely replaced ISA for use in personal computers. PCI use the same mechanical interface as ISA and is not suited for industrial use.

VME Is an official bus standard for I/O, memory and CPU modules [VITA]. Logically it is closely related to the Motorola 68000 bus, but has very high standards for signal integrity.

Mechanically, VME is built over the standard EURO rack system, supporting the 3U and 6U form factors.

VME is extremely robust both electronically and mechanically, and is well suited for industrial applications. The main drawbacks of VME systems are their high price, high power consumption and large space requirements.

Compact PCI Is electrically similar to PCI, but is built over a standard EURO rack system similar to VME [PICMG]. Compact PCI supports 3U and 6U form factors with the same mechanical reliability as VME bus.

While Compact PCI modules are cheaper and less power hungry than VME modules, they have the same requirements for space.

PC 104 Is electrically identical to ISA, but build over a more robust mechanical construction, replacing the traditional ISA connectors with stackable pin-headers [PC/104]. PC-104 modules are typically 90×95 mm, but modules with other dimensions are common. Each board occupies 15mm of vertical space when stacked.

The use of stackable pin headers to carry the bus, makes it possible to build quite compact systems as little space is wasted for interconnections. As strict mechanical requirements are not enforced, PC-104 is prone to impractical mechanical solutions, especially regarding connections to the outside world.

Industry Pack or IP is an official standard for industrial I/O and memory modules. IP modules are so called mezzanine modules, intended as board-level rather than rack-level modules [Mezz]. As such, they provide a hybrid between board-integrated I/O and fully modular systems like PCI or VME. IP connectors can be placed directly on CPU modules, letting IP modules replace closely integrated I/O circuitry.

As IP modules are intended for board-level integration, the connector toward the outer world is defined by the standard, making it necessary to introduce an external *transition module*, that interfaces between the standard IP connector and whatever connector is intended for the external interface.

IP modules effectively separate the parts of the physical interface of figure 5.2. The Canonical part is implemented on the IP module itself, while the mechanical part is implemented on a separate *transition module*. The signal conditioning can be implemented on either or both modules, as the designer pleases.

While many other modular I/O systems exist, the ones described above are representative of the current market for modular I/O systems with close integration between the CPU and the I/O modules.

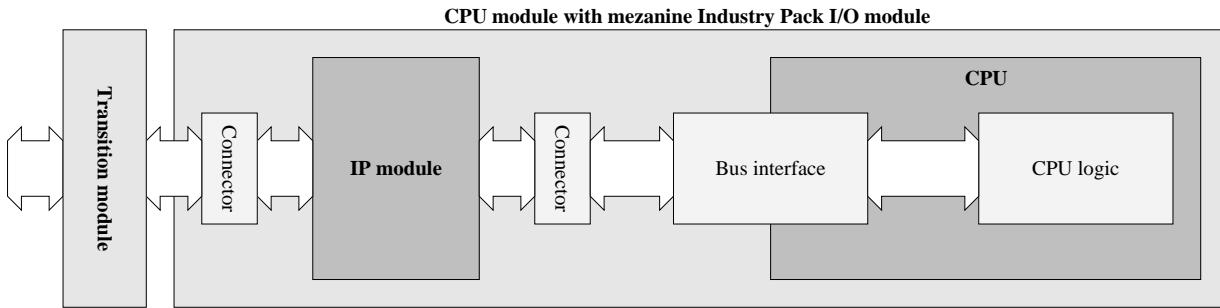


Figure 5.4: Typical IP module application

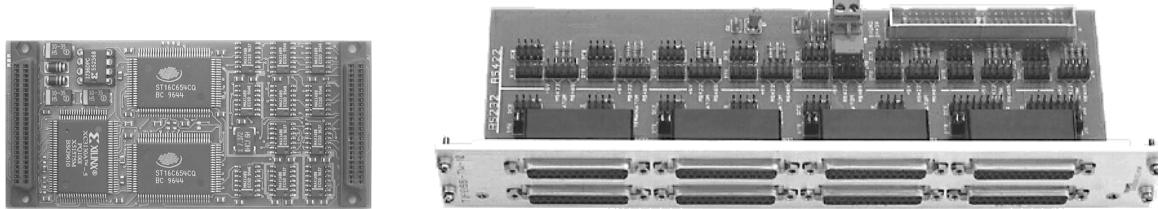


Figure 5.5: Typical IP module with transition module (*not in same scale*)

Other I/O systems

While using a traditional parallel bus as an interface for modular I/O is still the dominant technology, other technologies have evolved, to support applications where the traditional method is impractical.

Board-level serial busses: A large range of different micro controllers exists for embedded applications, but it is often necessary to expand their build-in I/O capabilities with external I/O devices. In order to maintain a high degree of integration and low cost, several serial busses, like I²C and SPI have evolved to replace the traditional parallel busses. The serial busses serve the same purpose as parallel busses, but requires fewer components, less space and design effort, at the cost of bandwidth.

As there exists a large number of I/O IC's supporting board-level serial busses, it is perfectly feasible to design a modular I/O system like IP around them, but in practice they are primarily used in static designs for embedded applications, requiring a high level of integration.

Distributed I/O: The close integration between CPU and I/O offered by the bus systems discussed above is impractical for applications where the resources controlled or monitored by the I/O is not placed close to the computer. A typical example of this is factory automation.

I/O systems based on industrial LAN's like ARC-net or CAN-bus have evolved to avoid the extensive wiring and signal integrity problems associated with centralized I/O systems.

In distributed I/O systems, each I/O module is an autonomous computer with integrated I/O functions. In essence, the board- or rack-level bus system is replaced with a LAN, that offers the appropriate signal integrity and bandwidth.

5.2 Reconfigurable technology

Instead of using physically exchangeable components in the form of I/O modules, we have developed an I/O technology, based on configurable logic networks, that achieves its flexibility from the fact that the functionality of the logical network can be configured for a wide range of applications.

We use a *Field Programmable Gate Array* (FPGA) connected to the CPU bus to implement the bus interface, as well as the I/O logic. In digital I/O applications, the FPGA also implements the canonical part of the physical interface, and may even implement the signal conditioning part if there are no critical demands for buffering etc.

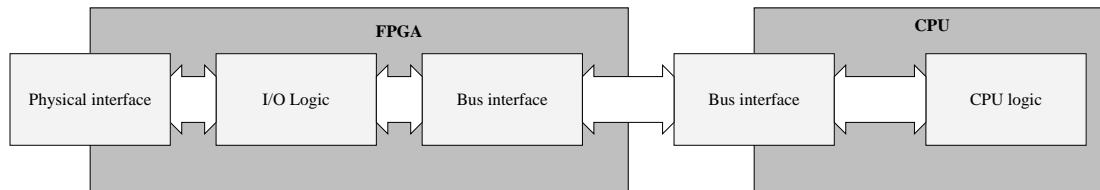


Figure 5.6: Block diagram of FPGA based I/O system

FPGA's

A FPGA is basically an array of logic cells, that can be configured to perform simple combinatorial or sequential operations. A simple example of a FPGA cell is shown in figure 5.7. The lookup table can be configured for any combinatorial function, mapping four inputs to one output, while the multiplexer can be configured to pass the registered or unregistered output of the lookup table.

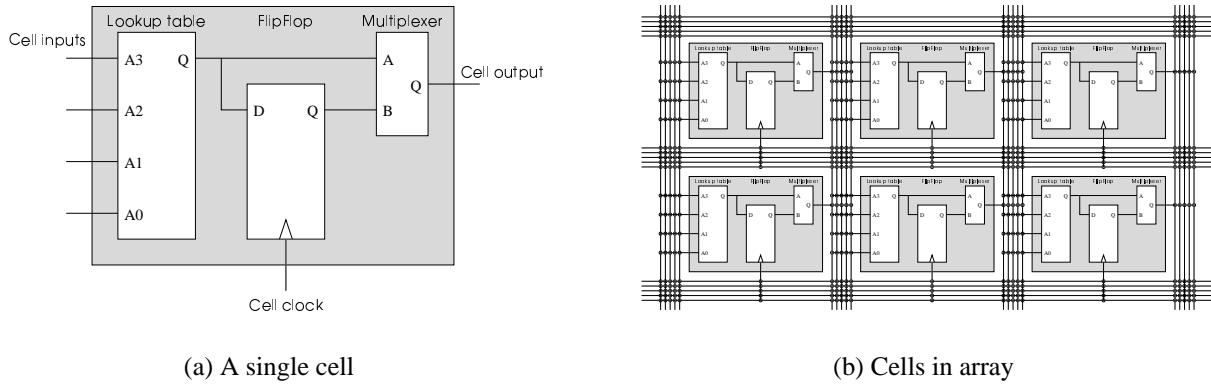


Figure 5.7: Simplified example of FPGA structure

By placing a large number of simple cells in an array, with reconfigurable interconnections, it becomes possible to configure the system to perform arbitrary logical functions, limited only by the number of available cells and interconnections.

Practical FPGA's have hundreds or thousands of logic cells, allowing them to implement functions ranging from simple boolean operators, over memory blocks, to CPU's and other complex state-machines. As the cells perform their operations in parallel, FPGA's can implement many individual functions at the same time. The massive parallelism of the FPGA makes it possible to avoid the traditional bottlenecks of the CPU, making it possible for FPGA's to outperform software implementations of a large range of simple and repetitive algorithms.

The internal signals of the FPGA is in contact with the external world through a number of specialized I/O cells. Each I/O cell controls an I/O pin on the FPGA IC, and can typically be configured for Input, Output or bidirectional operation.

The FPGA's used during this project have been the:

Xilinx XC4010XL: The entire XC4000 series have logic cells (CLB's) with a modest complexity. [XC4000]. The XC4010XL have a total of 400 CLB's, organised in an array of 20×20 , giving it a total complexity equivalent to 10,000 gates.[XC4000]

Xilinx XC4020XLA: Have a total of 728 CLB's organized in a 28×28 array, giving it a total complexity equivalent to 20,000 gates.[XC4000]

Xilinx XC4044XLA: 1600 CLB's organized in a 40×40 array, giving it a total complexity equivalent to 44,000 gates.[XC4000]

Xilinx XC2S200: The entire XC2S series (Spartan II) have more complex and flexible CLB's than the XC4000 series, increasing the functionality of each CLB dramatically. In addition, the Spartan II series have on board RAM, which frees a lot of CLB's in applications requiring memory registers. The XC2S200 have 1176 CLB's and 56K bits of block RAM, giving an overall system complexity equivalent of 200,000 gates [XC2S]

All FPGA's used in this project have their configuration data stored in configuration RAM cells, making it necessary for the FPGA's to read their configuration data from an external PROM at each power-on.

Specification and programming

FPGA's are much too complicated to be programmed manually cell by cell using boolean algebra. Instead, we use development tools that allow us to specify the functionality in a high level specification language, and then transform the specification into low level configuration data for the FPGA in question.

One of the main benefits of this approach is our ability to reuse the high level specifications on other FPGA's, allowing us to move across different FPGA platforms quite easily.

One of the main drawbacks of this approach is that we separate our knowledge of the internal FPGA architecture from the functional specification, leaving performance optimization to the development software. If the full performance potential of the FPGA is to be utilized, we need to apply our knowledge of the FPGA architecture at some point of the development process. Experience shows that when demands for performance and complexity increases, substantially more time is used in *assisting* the development tools to implement a design for a specific FPGA, than in specifying the functionality.

While the use of high level specification languages does not remove the hassle of meddling with platform specific details, when high performance is involved, our experience shows that it is an excellent tool when flexibility is more important than performance.

We use VHDL as specification language, and use the *Foundation ISE* integrated development tools from Xilinx to transform our high level designs into configuration data for the devices. VHDL is very similar to programming languages used for software, as it allows the source code to be broken down into modules with well specified interfaces, that can be linked together.

Architecture of I/O subsystem

The architecture of our FPGA based I/O system is equivalent to a traditional I/O system, as we use VHDL modules in the place of physical modules, and we connect the VHDL I/O modules to a common CPU interface module. As shown in figure 5.8, a number of I/O modules are defined in VHDL and *connected* to the CPU interface which is also defined in VHDL. While the interface between VHDL modules are specified as part of the modules, the physical connections between the VHDL modules and the exterior of the FPGA goes through the I/O cells of the FPGA, which can not be completely specified in VHDL.

The hierarchy of the specification modules are shown in figure 5.9. The top level module is written in VHDL, and defines the infrastructure of the system. It specifies the signals going

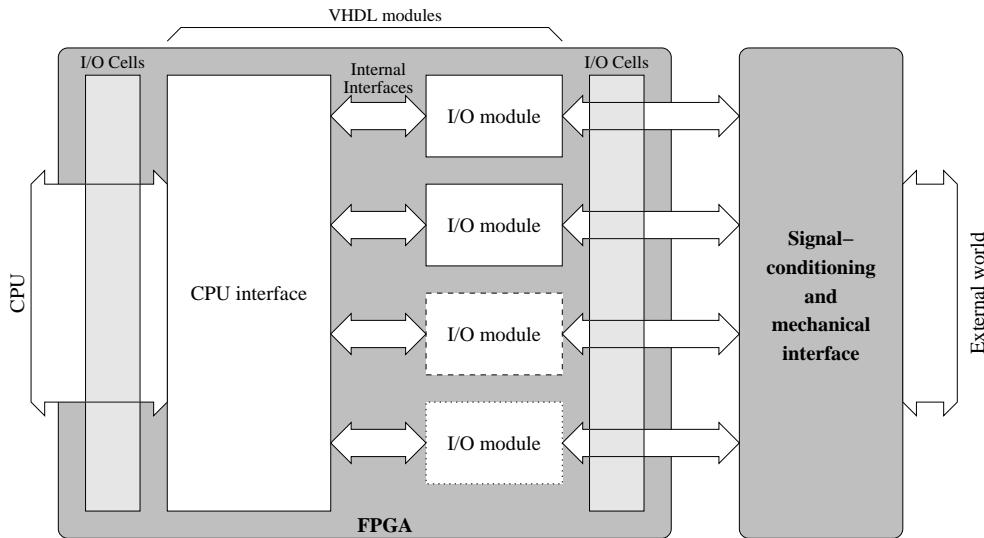


Figure 5.8: Architecture of FPGA/VHDL based I/O system

in and out of the system, and defines the internal structure of the I/O system by including and connecting the appropriate modules.

The top level module does not specify the physical properties of the signals going in and out of the FPGA. These are specified by the constraints *module* or -file. The constraints file specify properties like which pin to allocate for which signal, output slew rate, pull up/down resistors, timing constraints etc. It is of course highly device specific and must be adapted to each device we use for the I/O system.

The top level design includes a number of VHDL modules, defining the CPU interface and the I/O modules needed to perform the needed functions. All these modules are defined in VHDL, and may themselves include modules for sub components. While we wish to stay as device independent as possible by using pure VHDL, the benefits of including a device specific sub component can be so overwhelming that we are tempted or forced to do so even if it means re-implementing that part of the design when moving to other FPGA platforms. This is typically true for memory blocks, where we can typically reduce the CLB count with a factor of 16 or more if we use device specific sub components.

The relation between the different components and levels in the design hierarchy are illustrated in figure 5.10, showing the physical signal properties being defined by the constraints and the logical properties by the top level VHDL module. The top level module also defines a suitable infrastructure to connect the CPU interface to the I/O modules.

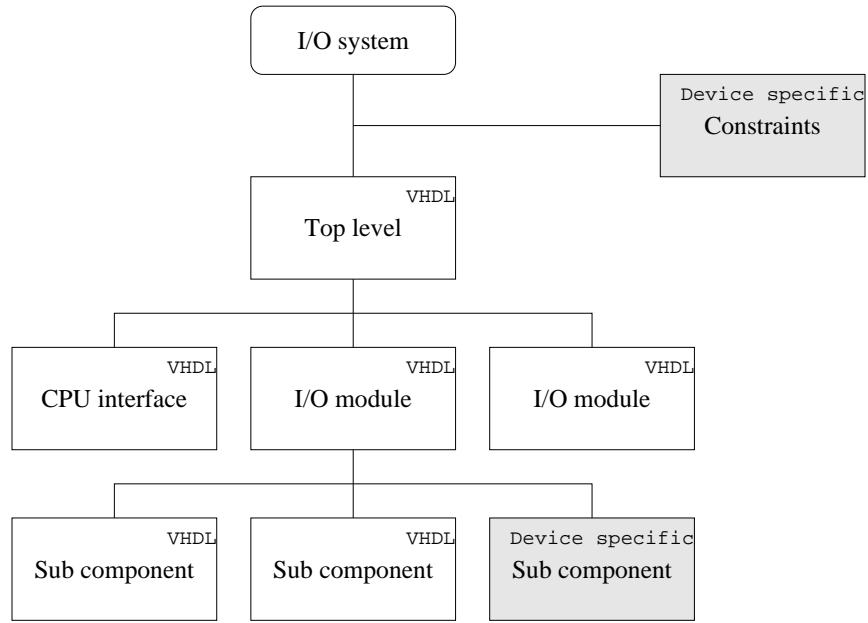


Figure 5.9: Design hierarchy of I/O system

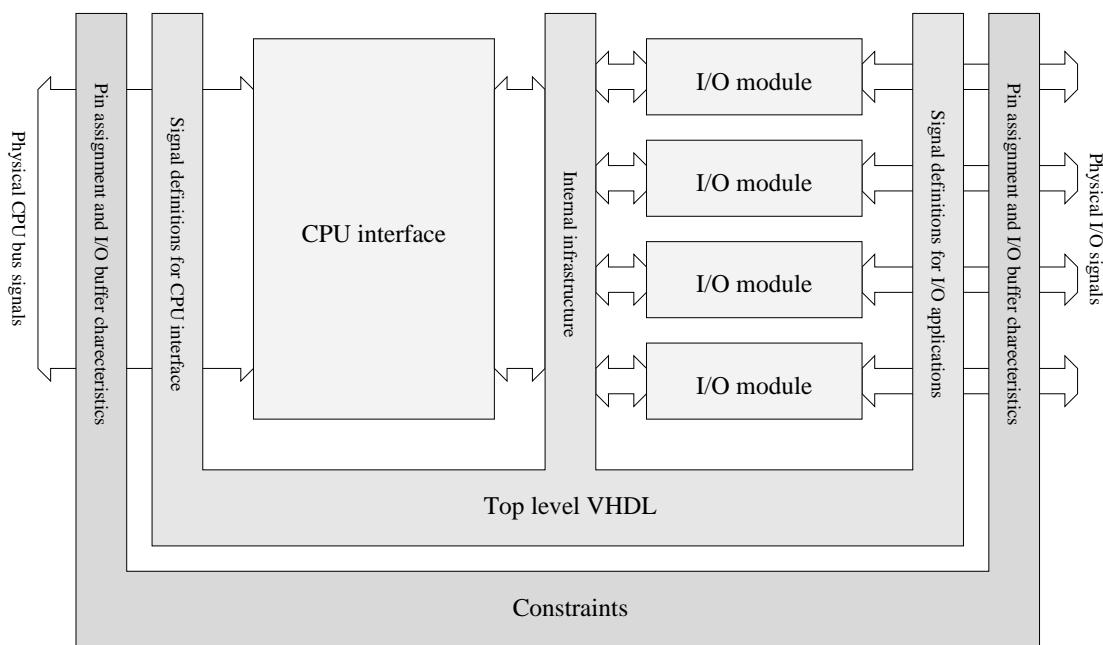


Figure 5.10: Layered architecture of I/O system

Simulation

A substantial benefit of the modularity of the VHDL specification is that each module or component can be simulated in order to establish its correctness prior to implementing the whole system. This is done by defining a *test bench* in VHDL. The test bench will submit the module under test to a range of stimuli and verify the response. The stimuli will typically be a simulation of the signals the real module will encounter under typical and extreme operation.

Simple simulation tools will not take device delays into account, but are still useful in order to establish the logical correctness of the design. More sophisticated tools will simulate the internal signal delays in the FPGA, and verify the correctness of the design with respect to specified setup and hold times.

Unfortunately we are not in possession of a simulation tool that will allow us to simulate the number of signals we need for our modules, ruling out this valuable method of verification and debugging. Fortunately our designs are simple enough that we can see through logical errors without simulation, and that we can verify and debug timing by external physical means. This approach is a usable practical alternative to simulation, but does not allow us to vary the test conditions from the typical situation in our physical setup.

5.3 Hardware platform

During the course of this project, we have implemented our I/O system on a number of hardware platforms, for 3 different CPU interfaces.

Platforms for the Microline bus

Our main objective is to supply an I/O system to the Microline bus, as we use CPU's with this bus interface to control our DT-VGT robots. In order to accomplish this, in the face of tight demands for size, we have developed 3 different FPGA based I/O boards for the Microline bus:

16 bit Test board: Using a piece of prototyping PCB, a PLCC version of a XC4010XL were connected to the relevant signals of the Microline bus, using prototyping wire. Using layers of copper and polycarbonate tape, we were able to supply HF stable ground and power supply to the package, ensuring a stable test board, that allowed us to verify and improve our basic design ideas. The PLCC version of the FPGA did not have enough pins to support a full 32-bit bus, so only 16 bits were used on this first version. The digital I/O signals of this board were available on a 25 pin sub-D connector, enabling signal conditioning HW to be implemented on other boards.

32 bit prototype board: By designing a 2-layer PCB, a TQFP-144 version of a XC4010XL were connected to the Microline bus, using the entire 32 bit data bus. This prototype board

use a 68 pin SCSI-III connector to allow easy and reliable connection of 34 high speed digital I/O signals to external boards, by way of ribbon cable. Great care was taken in the HF aspects of the PCB design, and the board have successfully been used at speeds of 80MHz.

The board is compatible with all XC4000 series FPGA's with a TQFP or PQFP-144 footprint, and have been used successfully with XC4010XL, XC4020XLA and XC4044XLA devices. The board have been used for a range of projects and applications, ranging from control of a DT-VGT robot, to control of an autonomous guided vehicle, to high speed data acquisition from an ultrasonic scanner.

32 bit mother board: The pressure to make the control unit for the DT-VGT ever smaller, led to the decision to design a new board for the Microline bus, combining the Microline power supply, the FPGA based I/O and the ARC-net network interface to connect to the central controller. The new board was designed to allow a more compact - cable less - connection between the FPGA (mother) board, and the board supplying the signal conditioning HW (daughter board). The two boards are now connected by two vertical connectors, allowing the board to be stacked directly on top of each other.

The combination of PSU and I/O board, in combination with the space saving connection between FPGA- and daughter-board is a very space efficient solution, allowing a complete system to be build with a stack of 3 boards measuring only $14 \times 7 \times 5$ cm.

I/O for other CPU's

Our involvement in projects beside the DT-VGT have led us to interface to two other busses:

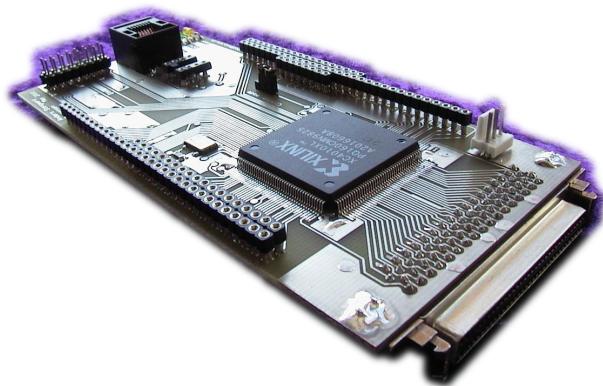
Industry Pack: In order to interface a VME computer with support for Industry Packs, to the motors and wheels of an electric vehicle¹, we mounted a PLCC version of a XC4010 on an IP prototyping PCB.

PCI bus: As an opportunity to gain knowledge of the PCI bus, we defined and supervised a final project from Odense Engineering College, porting our I/O system to a Spartan-II based experimentation board for the PCI bus [Knudsen01].

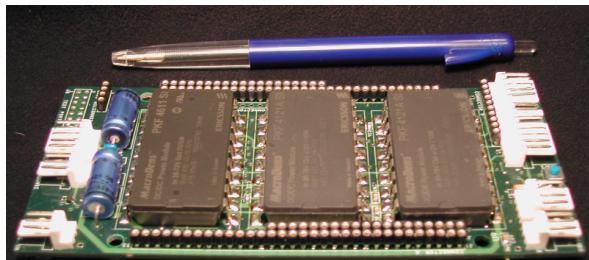
5.4 Internal architecture

The architecture of the system is a reflection of our main objectives and the technological possibilities.

¹An 'Ellert'



(a) 32-bit prototype board



(b) 32-bit mother board — PSU side



(c) 32-bit mother board — FPGA side

Figure 5.11: FPGA I/O boards for Microline bus

Objectives

Our objective is to develop an I/O system that can supply all the types of I/O needed for robot control, to a large range of computer platforms or computer busses, including Microline, Industry Pack, VME, ISA and PCI.

To demonstrate the system, we will implement an I/O system for the Orsys Microline bus, interfacing it to the I/O necessary to control a hydraulic DT-VGT robot module.

The popular CPU busses/interfaces mentioned above are all very similar. They are all organised in 8, 16 or 32 bit words, discriminated with a binary addressing scheme. The main differences lie in the bus width, the addressing modes, the timing and the synchronisation. By choosing an internal architecture that is similar to the popular busses, we can later develop efficient interfaces between our I/O subsystem and the various external CPU interfaces. It will also be possible to interface to a range of other parallel and serial CPU interfaces.

Based on our experience with robot control, we know that the most likely I/O interfaces will be: 8 – 16 bits analog inputs and outputs, digital inputs and outputs, quadrature encoder inputs, pulse width modulated (PWM) outputs and an array of 8-bit field busses, including e.g. CAN-bus, ARC-net, and SERCOS.

Registers and addressing

We choose a word width of 32 bits, as this is directly compatible to most modern CPU's, and is sufficient for all I/O applications we have previously encountered. The 32 bit architecture can be represented as an array of smaller words if we interface to architectures with smaller word lengths, and as a fraction of a word if we interface to architectures with larger word lengths.

In order to make the system easy to understand and work with, we have chosen a simple and traditional asynchronously shared bus architecture to link the I/O modules and the CPU interface. The internal bus is inspired from VME bus, which we have a lot of experience with. In the clarity of hindsight, we should have chosen a different architecture, as the traditional shared bus offers poor performance when implemented in a FPGA. In fact, converting our design to a more optimal architecture has a high priority among our future work. We find that the open interface *WISHBONE* [Wishbone] is a very promising alternative for this line of research, that should be investigated further.

Each I/O module will be represented by a number of 32-bit registers, placed within one or more blocks of the systems total address space. We chose a 16 bit address bus, giving the system a total capacity of 2^{16} 32 bit registers. The decoding of the 16-bit system address bus is local to each I/O module.

Only two operations are allowed on the registers: *read* and *write*. Each operation is signalled by the CPU interface, by asserting a **read** or **write** signal. The CPU interface may be implemented

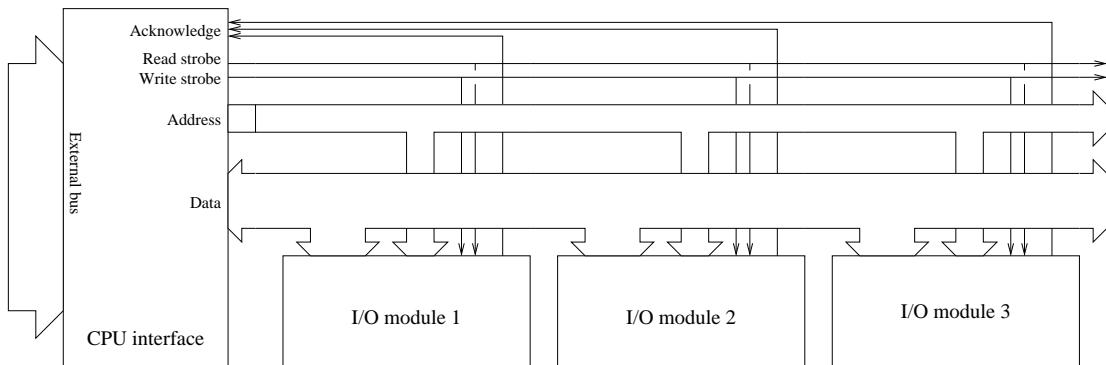


Figure 5.12: Internal architecture

in such a way that other external operations, such as block read or write are supported towards the external CPU.

As the amount of time needed for read and write operations may vary between the I/O modules, and as we do not know the speed of the external CPU, we need a mechanism to control the speed of the internal data flow. Each I/O module will assert an acknowledge signal when a read or write operation is successfully completed. The CPU interface will wait for this *data transfer acknowledge* signal in order to finish the corresponding external data transfer, forcing the external CPU to generate wait-states if necessary.

5.5 I/O modules

During the course of this project, our FPGA based I/O have been used for a variety of applications, primarily for robotics and industrial control, but we have also contributed to other areas. A list of known applications are given below:

- Read Only Memory
- 8 bit binary output
- 2 digit (8 bit) hexadecimal display
- 8 digit (32 bit) hexadecimal display
- SPI interface for 8 channel ADC
- SPI/CAN interface for Temposonic sensor
- Various PWM generators
- IRQ controller
- Dual power bridge controller
- Quadrature counters for encoders
- Ramp generator with integral delta-sigma DAC
- 10-bit 60MHz peak detector with RAM buffer
- Dot matrix display controller
- Interface for Play station control pad
- Endian converter

Evidently, some of the modules, like the ROM module, has no external connections, and is not really an I/O module. The same is true for our endian converter and IRQ controller. As they have all been implemented to support various I/O applications we find it relevant to mention them all the same.

Structure

The I/O modules are all designed over the same basic structure, shown in figure 5.13.

The memory may simply be a set of latches, or an assembly of block memory e.g. dual ported RAM. It is controlled partly by the internal bus, via the bus control logic, and partly by the I/O logic.

The bus control logic contains an address decoder, and the necessary logic to allow the internal bus to access the memory in a way that is consistent with both the bus specifications, and the purpose of the I/O module. Bus events can be signalled to the I/O logic.

The I/O logic interfaces the external world to the content of the memory. This may be achieved by direct representation, as in a digital I/O application, or by more or less complicated sequential operations. It might utilize clock signals from the top level design, and it may also export IRQ signals to an IRQ controller through the top level design.

Evidently, modules without direct I/O functionality, like ROM modules, does not have external connections, and their internal logic, if any, should not be called I/O logic, but they are implemented over the same basic structure.

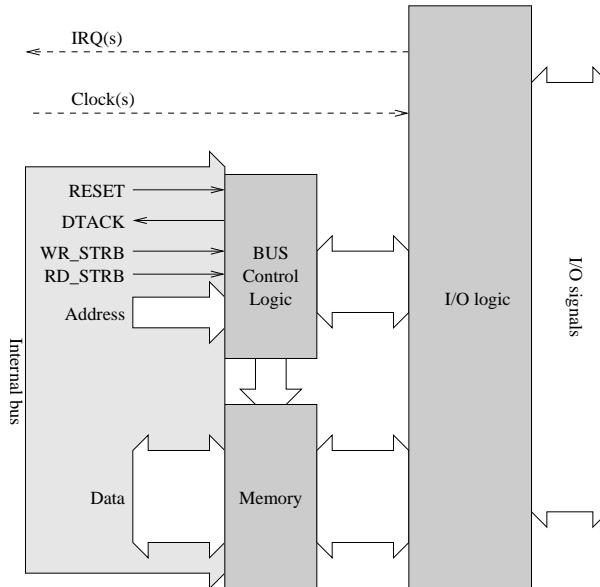


Figure 5.13: Basic structure of I/O modules

5.6 The CPU interface

Although we have implemented CPU interfaces for 3 different bus architectures, we will only discuss the interface for the Microline bus, as it is central with respect to our overall project.

The Microline® bus

The Microline bus which is defined in [Orsys-A], is a simple synchronous 32 bit bus, with a 40/50/60MHz bus clock. It is build upon the native I/O bus of the TMS320C32 DSP from Texas Instruments [TMS320C32], with the addition of a number of control signals. The Microline® bus is a company standard, developed by *Orsys GMBH*, as the peripheral bus for their DSP solutions.

The timing is slightly different from the original TMS320C32 specifications, as Orsys have inserted buffers and control logic in some data paths. Unfortunately the exact nature and magnitude of the differences are not described in the Orsys documentation, and we had to reverse engineer the CPU module in order to obtain usefull timing specifications [Sørensen-B].

As the I/O bandwidth requirements for the present applications are moderate, we have settled for a simple implementation of the bus interface, that typically requires 3 bus cycles for each I/O operation. A detailed description of our implementation exists in [Sørensen-C].

Discussion

While it seems like a simple task to design a CPU interface for a bus like Microline®, our experience show that the desire to achieve high performance complicates matter, as we have to make a design that can benefit from the typical timing characteristics, while still work under worst case conditions.

We were disappointed that it was not immediately possible to design a bus interface that allow us to complete bus operations in two clock cycles. The use of an asynchronous internal bus is one of the main problems in achieving two cycle operation, as we both need to synchronize the data transfer acknowledge to the external bus cycles, and provide a suitable delay to accommodate setup and hold times.

The bus drivers and control logic Orsys have inserted in front of the CPU is also a complication, as the signal delays caused by these components deteriorate the setup and hold margins in the bus timing. Our observations suggest that Orsys' specifications for worst case signal delay through their control logic are far too conservative compared to typical delays. If this suggestion can be verified, we could probably cut one clock cycle off I/O operations. The same benefit could be obtained by abandoning the Orsys board in favor of an integrated DSP/FPGA module.

The current CPU interface was designed at a time where our experience with VHDL and the Microline® bus were limited. We feel that both performance and clarity might benefit from a redesign of the whole CPU interface, but are reluctant to commit the necessary resources until we can redesign the whole internal communication scheme.

5.7 Conclusion

Much of our efford during this project have gone into the generic I/O system, as we feel that this technology is one of the primary factors that will enable us to interface our generic embedded control nodes with practically any robotics module.

The generic I/O technology have lived up to, and superseeded, our initial expectations, as it has proven itself to be a very powerfull tool when designing, developing and integrating embedded control technology.

The advantages of the technolgy can be summed up as:

- The flexibility of the technology ensures an extremely broad area of compatibility, enabling us to interface to virtually any external electronic system.
- The speed and massive parallelism inherent in the technology ensures a very high I/O bandwidth, that exceeds any bandwidth requirement we have encountered in industrial control systems.
- The flexibility of the technology makes it ideal for *rapid prototyping*, which makes it a very attractive platfrom for research and development that involves I/O.
- The specification of I/O components can migrate between different computer platforms, making the technology very interesting for research and development involving heterogeneous computer platforms.

Although we have not chosen the most optimal way to implement our technology, we have developed a completely operational I/O system, that satisfies and exceeds our requirements with respect to flexibility, performance and size. Additionally, our implementation is flexible enough to allow external projects to benefit from *rapid prototyping* solutions based on our I/O system.

5.8 Future work

The prospects of this type of I/O are so interesting that we recommend it as an individual field of interest for the COmputer Systems Engineering² (COSE) research at SDU. During (and parallel with) this project we have already shown it's *rapid prototyping* benefits, dramatically reducing development times for computer interfaces, as well as it's potential for bridging the compatibility gap between various computer platforms. The great potential — as well as emerging commercial trends — lead us to believe that this technology will become quite common in I/O applications within the next few yars.

We reccomend a continuation of our work, with immediate attention to the following areas:

²Dataeknologi in Danish

- Complete redefinition of the internal architecture, to make it compliant with an open interface specification such as e.g. *WISHBONE* [Wishbone], and to utilize the inherent architecture of the FPGA better.
- Contemplation of the interface between FPGA and external signal conditioning, in order to suggest design guidelines that will facilitate compatibility between future applications. It might prove useful to consider emerging product standards like DIME and DIME-II [Dime] for future applications.
- Research of the possibilities presented by allowing the host CPU to reconfigure the FPGA dynamically, allowing the *hardware* specification to become entangled with the interface software.
- Investigation of alternative specification languages, with different levels of abstraction from VHDL, e.g. *Handel-C* [Handel-C]

Chapter 6

Network

The choice of network technology is, in many ways, one of the most critical decisions of this project. Any of the generic embedded control nodes (GEECONs), as well as the central control node can be transferred to a different technology, without affecting the other nodes, as long as the network and protocols remain unchanged or at least compatible. Changing the network technology, on the other hand, implies changes to every node of the system.

6.1 Overall network demands

In chapter 3, we established that the network technology should be as fast as possible, and at least support isochronous data streams with a bandwidth of 2Mbps and a worst case latency below $500\mu s$.

In addition to these demands, the technology must pose the following characteristics:

- **Reliable operation during continuous use in heavy industrial environments.** This property relies on signal integrity as well as mechanical robustness. As the prototypes developed for this project is only meant as a demonstration, we are allowed to save time and resources by implementing the network technology in a less robust way, as long as the technology itself does not inhibit later upgrades to industrial standards.
- **Support for asynchronous data.** As we need to pass commands, status information, parameters etc. between the central controller and the GEECONs, the network must support the transmission of asynchronous data parallel to the isochronous joint- and feedback-streams.
- **Network configuration.** As the network topology will reflect the topology of the modular robot, it is necessary for the central controller to be able to determine the network topology, if the system is to support automatic configuration of the central components of the robot controller.

- **Compatibility.** In order to encourage further development of our technology, we should chose a network technology that is well documented, widely accepted and in general use.

6.2 Network candidates

The sheer amount of different network technologies in existence, puts it outside our scope to make a comprehensive survey of them all.

At the beginning of the project we chose to investigate Firewire as our primary candidate, with an option to revert to ARC-net, with which we had previous experience.

Many other networks have been considered in more or less detail. Below we have commented some of the noteworthy, either because of their popularity, or their applicability to our work.

Ethernet

The synergy between popularity, development and product availability have ensured Ethernet a position as the local area network of choice for commercial and personal applications. Ethernet has also managed to established itself as a reliable solution for a wide variety of industrial applications.

One of the major drawbacks of using Ethernet for industrial applications, is that it is inherently not intended for real-time applications, as it does not support limited latencies.

It is possible to bypass the inherent lack of real time support, by overlaying the native protocols, with higher levels of real-time protocols. Such solutions will diminish the available bandwidth, and add complexity to the implementations.

The reduction of effective bandwidth may not be a problem, as Ethernet implementations with support for transfer rates above 1Gbps exist.

As real-time patches for Ethernet are not nearly as wide spread as networks with inherent real-time support, we find that Ethernet is inappropriate for this project.

Bluetooth

We have considered Bluetooth very briefly, as it is intended for highly reconfigurable systems. Unfortunately, it can be immediately discounted on accord of its limited bandwidth, and the potential problems of operating a radio based system in an industrial environment, especially at shipyards and other facilities where robots operate inside narrow and complex metallic structures.

CAN-bus

CAN, or Controller Area Network [CAN], originates from the car industry, as the internal network for automobiles. As such, it has many of the characteristics we require, including reliability and real-time performance. Unfortunately, the bandwidth is only 1Mbps, which is not nearly enough to support our demands.

Sercos

Sercos [SERCOS] is specifically designed for industrial motion control. It is a ring network based on optical fibers, and supports bandwidths of up to 16Mbps. Sercos support for isochronous streams exceeds our demands by a wide margin, but its support for asynchronous data is inflexible. As Sercos does not have any applications outside motion control, it will probably remain a narrow and virtually unknown network technology.

While Sercos is certainly attractive from a technological point of view, we fear that it may create compatibility issues with future partners, and we will prefer to use a more widespread technology.

USB

The Universal Serial Bus [USB] is one of the most popular network technologies for commercial and personal computer peripherals. USB is designed for modular technology, and USB-2 is especially interesting to us, with its inherent support for parallel isochronous and asynchronous communication at 480Mbps.

As USB is not designed for industrial applications, we are reluctant to accept it as a serious candidate, before it has been generally accepted by the industrial community.

WorldFIP (EN-50170)

WorldFIP [WorldFIP] is a relatively slow (up to 5Mbps) field bus technology, which have some interesting features compared to the more popular field busses. It supports parallel isochronous and asynchronous data channels, and it supports redundant (double) cable connections for increased reliability.

While WorldFIP seems technically superior, the very limited product range suggest that it is not very popular, which may cause compatibility issues with external partners.

6.3 IEEE-1394 — Firewire

Firewire is specifically designed to carry real-time high quality video and audio streams in isochronous channels, with parallel asynchronous data transfer [IEEE-1394] [Anderson99]. It currently supports transfer rates of 100,200,400 and 800Mbps. While it is intended for the consumer and professional A/V market, it has also been incorporated in industrial motion control by several different companies, among which Nyquist [Nyquist], is one of the most prominent examples. While the high availability of consumer products based on Firewire will ensure it's overall popularity, the rising interest in industrial Firewire applications is a significant indication of its acceptance as a reliable and robust network.

Overall architecture

Firewire is a point to point network, where each node effectively form a 3-port network switch, resulting in a *binary tree* network topology¹, which configuration can be determined by any node on the network.

Firewire derives its real-time performance from a time division multiplexing scheme, that operates at a fixed frequency of 8000Hz, equivalent to a period of $125\mu s$. In each period a certain amount of data can be transmitted, e.g. 5kB at 400Mbps. Up to 80% of this capacity can be allocated for isochronous channels, while the remaining 20% is reserved for asynchronous transfer.

Isochronous channels

When an isochronous channel of a certain bandwidth is allocated, it is calculated how large a fraction of each $125\mu s$ period the channel is going to occupy. If such a fraction is available, it is then reserved for that channel. Up to 32 isochronous channels can exist in parallel, placed after each other in the $125\mu s$ period. Only one node can transmit to an isochronous channel, but every node can listen in on the channel.

Asynchronous transfer

Data for asynchronous transfer is transmitted during the part of each $125\mu s$ periods not used for isochronous data. Rather than being package or stream oriented like e.g. TCP and UDP, the protocol for asynchronous transfer is memory oriented.

Each node on the network represents a virtual memory block that can be remotely accessed. Asynchronous data transfer then constitutes e.g. read or write operations across the network.

¹The connections may not form cyclic graphs

Each node can map a segment of its physical memory into the virtual memory map of the network, using DMA or other techniques, depending on the local implementation. When the memory map of a node is accessed, the CPU is notified that a remote operation has taken place.

Hardware architecture

Firewire carries its information using 2 differential signals, implemented as 4 wires (2 twisted pairs). Cable length is limited to 4.5m, with possibilities for extensions to 70m using optical fibers and transceivers. In most cases, a 6 wire cable/connector is used to allow integral power distribution to remote nodes, such as e.g. repeaters, switches etc.

Previously, most Firewire implementations relied on a dual, galvanic isolated IC chip set, where one IC implement the Physical layer, and the other implemented the link layer. Quite predictably, most suppliers have now begun to integrate the two functions in a single IC.

Even when using commercially available chip sets, the design of a Firewire interface is a complicated process, with some design issues outside our experience. As a consequence we have chosen to rely on commercially available Firewire interfaces, and our choice of the Microline CPU platform for our GEECONs, was influenced by the fact that a Firewire interface was available for the Microline bus.

Software support

While the lower protocol layers are conveniently implemented in hardware, some higher layer functions must be implemented in software like device drivers, interrupt service routines, and function libraries.

Fortunately, software support exist for the Microline products, as well as for the PC based operating systems considered as basis for the central controller node, i.e. Linux and Windows.

Applicability

Firewire is close to being optimal for this project:

- Its bandwidth allows plenty of room for adding external sensor feedback in future applications.
- Its origin as a multi-media network makes it particularly easy to integrate cameras with the robot.
- Its 8000Hz time division multiplexing ensures worst case latencies below $250\mu s$.

- Its tree structure and automatic configuration is very supportive of our vision for modular reconfigurable robots.
- Its popularity increase the compatibility of our technology.

The main objections to Firewire are:

- The limited cable length (4.5 meters) may cause problems with mechanical modules that exceed this length. We don't expect the majority of robotic modules to be this large, and if the situation should arise, it can be remedied with the insertion of electrical or optical repeaters.
- The available components are not inherently industrial, but as experienced motion control companies are ready to adopt them, we are willing to use and evaluate the technology for our demonstration system.
- We have no previous experience with Firewire, nor has any of our usual partners. This implies an elevated level of technological risk to the project. We minimize the risk by selecting a well known technology as backup if Firewire should turn out to disappoint us.

Experience

In order to evaluate Firewire, we have run a number of experiments with isochronous and asynchronous transfer between GEECONs and a central computer.

The GEECONs used for testing consisted of the same Microline C32 DSP module used throughout this project, equipped with a Microline 100/200Mbps IEEE-1394 peripheral board. We used the IEEE-1394 software provided by Orsys to access the network from the GEECON[Orsys-B].

The central controller consisted of an ordinary office PC (400MHz Pentium system), running Mandrake Linux 8.2. It was equipped with a 100/200/400Mbps Unibrain PCI-Lynx IEEE-1394 interface. We used an experimental IEEE-1394 device driver, kindly provided by developers from the Linux community [Linux1394].

Our overall experience with Firewire on the central controller is very positive. It turned out that PCI-Lynx compatible interfaces are becoming obsolete, in favor of PCI-OHCI compatible interfaces, limiting the support and development effort for PCI-Lynx. We suggest using OHCI compatible boards for future applications. The experimental Linux driver caused a few problems, which were solved once a proved version of the driver was released and incorporated in the official Linux kernel releases.

During our experiments with the system, we learned that Orsys implementation of Firewire interface was very ineffective with respect to hardware as well as software.

The link layer IC used by Orsys is intended for small micro controller systems, and it features a number of grave limitations:

- It only supports a fraction of the theoretical bandwidth due to the size of its internal FIFO.
- Its internal FIFO only supports one channel, making it impossible to utilize more than one isochronous channel.
- While it operates with 32 bit internal registers, its interface to the 32-bit DSP only has 16 bits, making access to the interface grossly ineffective, and removing the benefits of DMA.
- While the FIFO can be accessed without I/O wait-states, accessing the control registers of the link layer IC generates 5-7 wait-states, reducing the efficiency of the system even further.

We could not obtain the source code for Orsys Firewire support software, but using a logic analyzer to monitor the transactions on the I/O bus, we were able to reverse engineer some of it, learning that the software was not very efficiently implemented either.

We were able to implement satisfactory isochronous and asynchronous communication between the central node and the GEECON, but as it turned out, approximately 80% of the CPU capacity was used to service the Firewire interface during communication.

Conclusion

Apart from the poor implementation of the Firewire components for the GEECON, our experience support the view that Firewire is an interesting technology for modular control.

Our assessment of the Orsys 100/200Mbps Firewire interface and the accompanying software, is that it is adequate for very simple applications and demonstrations, but unsuited to support this project.

The poor implementation of the Microline Firewire interface, regrettably forced us to abandon Firewire for this project, and revert to our backup technology.

As we have abandoned Firewire at an early stage of the project — before designing enclosures for the GEECONs — we were not able to submit it to realistic signal integrity- or other relevant tests, and we can not offer a first hand assessment of its applicability to industrial environments.

Future work

After we had abandoned Firewire and integrated ARC-net instead, Orsys GmbH. launched a new 100/200/400Mbps Firewire Microline module, with support for multiple isochronous channels and 32 bit access, as well as an external DMA channel for isochronous transfer.

We have had the opportunity to evaluate the new Firewire module and the accompanying software, and have concluded that they are sufficiently efficient to allow a Firewire based network for a modular controller.

If it becomes relevant to reconsider the ARC-net solution implemented in this project, we recommend taking a second look on Firewire, based on the new technology available from Orsys.

6.4 ARC-net

ARC-net is specifically intended for real time applications in industrial environments [ARCNET]. Its real time performance is derived from its token bus topology, which ensures that all nodes are granted network access in an even and deterministic way.

ARC-net supports information transfer using datagrams or packages of variable size, up to 508 bytes. Datagrams can either be broadcasted or transmitted to a specific node, and each node is allowed to transmit one package at a time in a round robin fashion.

Up to 255 nodes can be connected to the same network, and each node can determine the presence of other nodes. As the network is implemented as a bus, the relative position of nodes can not be determined without external means.

ARC-net can be implemented on a variety of physical medias, e.g. coaxial and twisted pair cables, with transfer rates up to 10Mbps.

Applicability

We find ARC-net to be an adequate technology for our distributed control system. We have selected it among other adequate candidates due to its popularity and our prior experience with it. The most interesting technical points regarding the use of ARC-net for this project are stated below:

- With a transfer rate of 10Mbps, ARC-net has sufficient bandwidth to support our worst case demand estimate for joint- and feed-back streams.
- According to our worst case estimates, up to 5000 bits of joint set point information and 5000 bits of feed-back information will be transmitted on the network within each period of the joint stream sample frequency. Transmitting 10000 bits will take at least $1ms$, and we must be careful to design the execution layer software to break the information down in suitable units in order to obtain acceptable latencies. In this respect it is important to note that the latency is the sum of transmission time and time spent waiting for the token.
- ARC-net interfaces are typically implemented using an integrated ARC-net controller IC that implements the entire ARC-net protocol, and connects to a CPU through an ordinary parallel peripheral bus interface. Such a controller can easily be interfaced to the CPU module used in this project. For the central control node, we can draw on the large selection of commercial ARC-net interfaces for ISA, PCI and other popular peripheral busses.

- A number of well proven industrial signal transmission technologies are available with ARC-net. We have no reason to doubt that a reliable and robust ARC-net implementation can be integrated with our system.
- ARC-net is inconvenient, for modular systems that include branches, as it is complicated to introduce branches on a bus oriented network medium. As we expect most aggregated robots to contain few, if any, branches, we are willing to accept this inconvenience.
- ARC-net does not immediately support automatic configuration, based on the topology of the system, as the individual position of network nodes can not be determined by default. This problem can be remedied by breaking the network medium into isolated segments, which are brought online one at a time during initialisation. This method have previously been used successfully with CAN-bus, in [Dalgaard01].

Experience

An ARC-net interface was actually developed before Firewire was abandoned, as we use ARC-net to interface to the PA-10 robot, described in chapter 8. The experience with interfacing ARC-net to the GEECON was positive, and the ARC-net interface was incorporated into the design of the GEECON motherboard², when Firewire was abandoned.

Experimental execution layer software have been developed and demonstrated by Micheal Bøllingtoft [Nielsen02] and Mads Lundstrøm [Lundstrøm01], for control of a PA-10 robot, and their experience with ARC-net communication between a PC, and a GEECON is encouraging, as they successfully controlled the PA-10 robot in accordance with an online joint stream being delivered from the PC to the GEECON via. ARC-net.

Based on experience from the experiments performed by Michael and Mads, execution layer software supporting a more general protocol is currently being developed for a demonstration of aggregated robots in conjunction with the EU project DockWelder.

Conclusion

We have incorporated an ARC-net interface in the generic embedded control nodes (GEECONs), and have verified that they can communicate with a central control node.

We have estimated that 10Mbps ARC-net is adequate to support the communication demands for our modular control system, provided the execution layer implementation takes the nature of ARC-net into account.

On line transmission of a joint stream through ARC-net have been demonstrated for a single GEECON and a central node, and generic execution layer software is currently being developed

²See chapter 4

based on experience from that demonstration.

6.5 Discussion

As stated in the beginning of this chapter, the choice of network technology is crucial to any distributed system design. We are happy with the decision to use ARC-net, as our previous experience with this technology will enable us to complete the distributed controller system with few surprises due to the network.

While ARC-net can support our demands for bandwidth and latency, care must be taken when designing protocols for isochronous and asynchronous data transfer, as ARC-net will not guarantee the required latency under all imaginable conditions. It is especially important to consider:

- The size, content and transmission method (node to node or broadcast) of individual ARC-net datagrams in relation to the token passing algorithm and the transmission speed of the network.
- The policy for retransmissions of corrupted datagrams, and it's influence on the latency.

We are disappointed that it was not possible for us to implement the system using Firewire. We still have a firm belief that Firewire is an excellent technology for modular control, and we hope we will be able to investigate Firewire further in future automation projects.

6.6 Conclusion

During this chapter, we have defined demands for the network technology to be used in the project, and chosen two network technologies that honor these demands.

Our primary candidate, Firewire, was evaluated during the project. Although we found it to be a very promising technology, we had to abandon it due to the lack of efficient Firewire implementations for our GEECONs.

To replace Firewire, we have integrated an ARC-net interface with the GEECONs. The interface have been tested and verified, and it has been established that the GEECONs can communicate with a central controller node and receive an on line joint stream.

Execution layer software that communicate online with a central controller have been demonstrated with a single GEECON.

6.7 Future work

The current execution layer software was intended for a simple demonstration, and the next step in this development is to implement more general execution layer software for the GEECONs as well as for the central controller. This work is currently in progress.

During the project, more efficient Firewire implementations have become available for the GEECONs. If the communication system of our distributed controller is ever to be redesigned, we recommend reevaluating Firewire as a candidate.

Chapter 7

DT-VGT interface

Using our generic embedded control nodes (GEECONs) with the DT-VGT is one of our major demonstration goals for this project. The demands for compactness and flexibility put on the controller by the DT-VGT have influenced many design decisions during the project, and we have allocated a lot of resources in order to bring this particular mechanical technology together with our GEECON.

7.1 The DT-VGT platform

DT-VGT stands for **D**ouble **T**ripod **V**ariable **G**eometry **T**russ. The moving element of the DT-VGT is a truss element, consisting of two inverted tripods, that can alter their shape under the influence of external actuators.

Truss elements have high strength to weight ratios, and the same is true of variable geometry trusses, which can typically carry more than their own weight. The DT-VGT's we are working with are designed to position and move a standard industrial robot arm inside a complex environment, like the inside of a ship hull. The advantage of this particular parallel mechanism is the combination of slender body, high strength, and large angular displacement, that makes it well suited to maneuver inside complex environments, rather like a snake.

The DT-VGT's are developed by the company Meganic, which is our close partner in this project.

Hydraulic actuators

The DT-VGT platform we are using, is powered by hydraulic actuators, in order to achieve the necessary strength. Each actuator connects from a leg in the lower tripod, to a cylindrical supporting base.

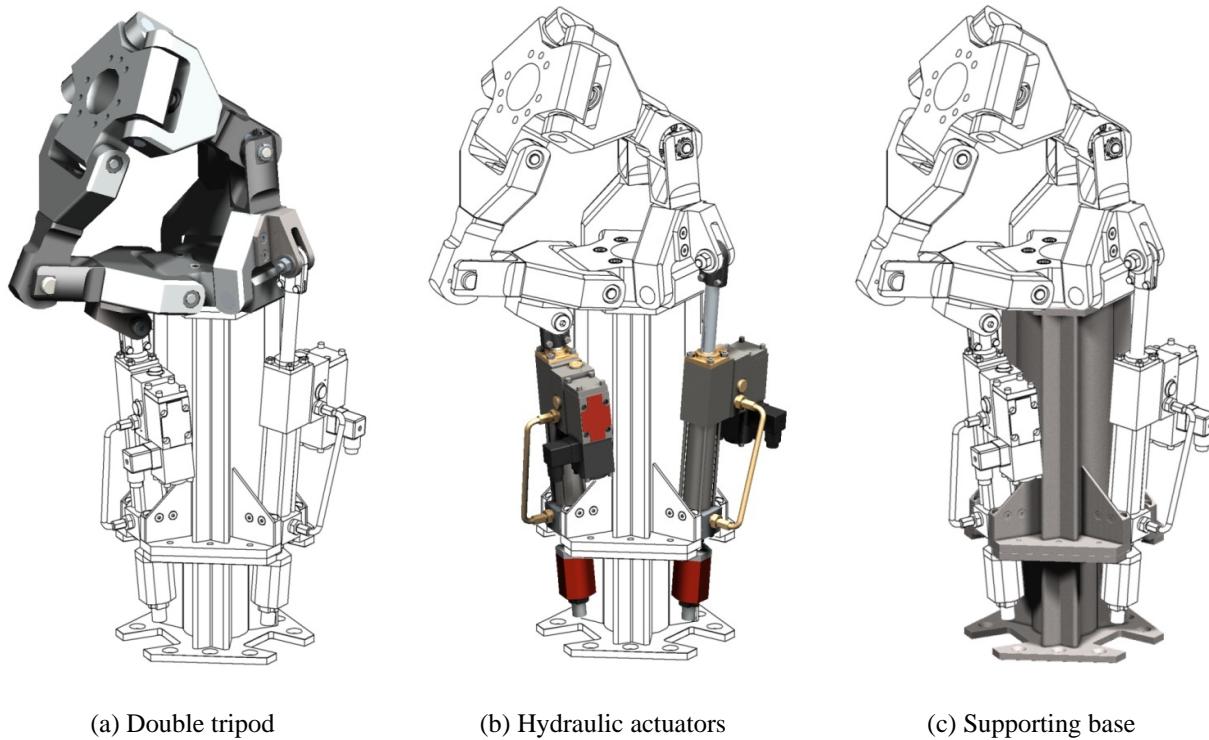


Figure 7.1: Breakdown of DT-VGT robotic module

Valves

The hydraulic actuators are controlled by proportional valves, which are unlinear and harder to control than servo valves. Proportional valves have been chosen in order to keep costs down, as Meganic hope to commercialize the DT-VGT. The valves are controlled by moving magnet electromagnetic motors, quite similar to the magnet/coil system of ordinary loud speakers.

We are in possession of a single valve which is functionally identical to the others, but feature spool position feedback. While spool feedback will probably enable increased control performance, the additional size and price of this feature means that we are only permitted to use spool feedback valves for laboratory experiments, not for the final system.

The linear motors used to drive the valve has a rating of 24V/4A, which is in reality a power rating of 96W, corresponding nicely with the coil resistance of approximately 6Ω .

Position sensors

In order to control the motion of each actuator, the hydraulic cylinders are fitted with internal position sensors, that measure the position of the piston with an accuracy of $5\mu m$ at a rate of

up to 2kHz. The sensors have integrated micro-controllers, and transmit their readings digitally through a CAN-bus network.

7.2 Switched mode valve amplifiers

A previous master thesis project, concerned with controlling an older VGT prototype, recommends increasing the reaction speed of the valves in order to obtain better control performance [Olsen00]

The reaction speed of the valve is defined by two factors:

- The available force to mass of the linear motor and its load, i.e. maximum current.
- The electrical induction to resistance in the linear motor, i.e. the rise time of the electric current.

The maximum current is defined by the ratio of available voltage to the electrical resistance of the motor coil, and can thus be controlled by the applied voltage. The rise time of the current is a constant ($\tau = L/R$), but the time it takes to rise from zero to a certain absolute current, is highly dependent on the applied voltage.

Though both factors determining reaction speed can be improved by raising the available voltage for the valve amplifier, care must be taken to control the current, as not to exceed the power rating for the valve. In principle, the dielectric strength of the coil wire isolation should also be considered if the voltage is raised substantially with respect to the rated voltage.

Intelligent amplifier

The task of developing the amplifier was sub contracted to an engineering student from *Ingeniørhøjskolen Københavns Teknikum* named *Jakob Lindeløv* [Lindeløv01]. He developed a small switched mode amplifier with the following specifications:

- 48V supply.
- 5A continuous current at 20kHz switch frequency without heat sink.
- CPLD controlled switch patterns.
- Programmable current limit from 0 to 8A.

All documentation of his work is contained in his final report [Lindeløv01], with comments and documentation of minor improvements in [Sørensen-A].

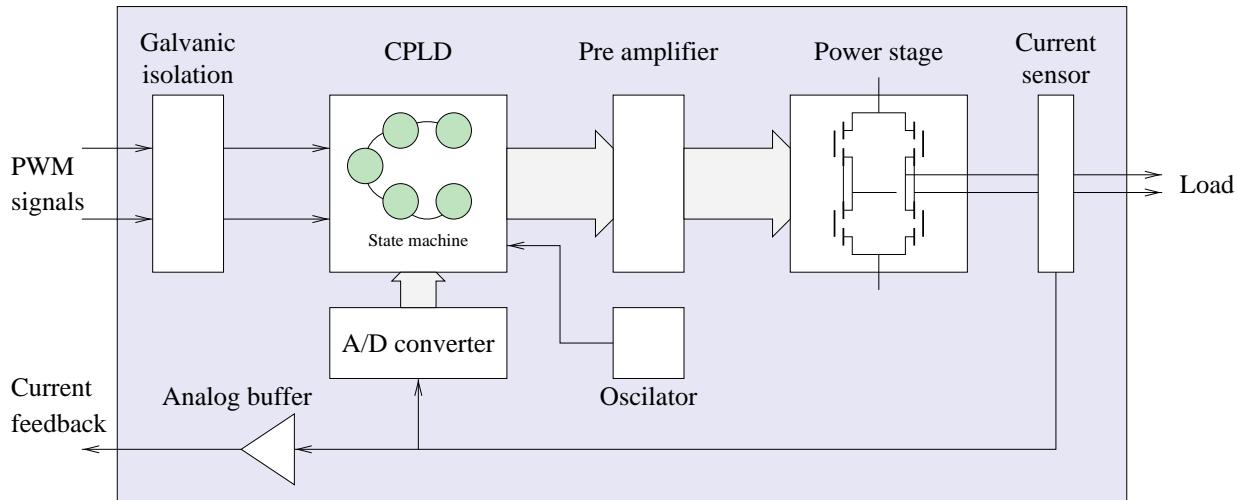


Figure 7.2: Block diagram of valve amplifier

Switch patterns

Jakob Lindeløv's design utilizes an ingenious switch pattern in order to obtain high efficiency and distribute heat losses between all 4 transistors in the power stage. The switch pattern is controlled by a simple state machine, implemented with programmable logic (CPLD). The design also features a current sensor and an A/D converter, that allows the state machine to limit the output current.

Miniaturization

While the switching patterns developed by Jakob Lindeløv allow us to avoid heat sinks, making a very efficient and compact power stage possible, Jakobs's implementation of the amplifiers is too large to fit into the allowed space of the DT-VGT controller enclosures.

Jakobs design was developed as a stand alone switched mode amplifier, but when we use it in conjunction with the FPGA based I/O system of our GEECONs, the I/O system makes the A/D converter and CPLD obsolete, allowing us to implement a miniaturized integrated 3-channel version of the amplifier, with only power stages, pre amplifiers, and current sensors, letting the A/D converter and FPGA of the GEECON control the switching patterns and current limits of the 3 amplifiers necessary to control a DT-VGT.

We have verified that it is possible to implement a sufficiently small integrated amplifier, but we have not yet designed it in detail.

7.3 Power supply

The total power needs of the DT-VGT electronics can be summarized as:

GEECON: 48V 0.2A

Valve amplifiers: 48V, 3A mean, 12A peak (to drive all 3 valves)

Position sensors: 24V, 0.6A

Resulting in a peak demand of 600W.

Due to cable and connector dimensions, it is impractical to distribute that amount of power at 48V, especially as we might wish to cascade multiple DT-VGT's in the future. We assume that we could reduce the peak current to the valve amplifiers without imposing grave compromises to the dynamics of the hydraulic control, but we will still need 2-300W of power for each DT-VGT, making a higher distribution voltage interesting, provided that we can use a sufficiently small, powerful and efficient voltage converter to be placed inside the DT-VGT.

We have chosen a solution where we reduce the peak current of the valve amplifiers to 3A each, use 300VDC for power distribution, and use ultra compact 500W DC-DC converters to generate 48V in each DT-VGT module. This reduces the peak current for each module to 1.7A, making it feasible to base the power distribution on ordinary $0.5 \dots 1mm^2$ distribution cable and compatible connectors. The 24VDC needed for the position sensors is generated from 48VDC by the auxiliary DC-DC converter on the GEECON mother board.

300VDC may seem unusual and impractical in other respects, but as most commercial and industrial electronics using 230VAC through a switched mode power supply will readily accept 300VDC, and 300VDC is easily obtained from a 230VAC supply, our power distribution scheme is in fact quite compatible with existing technology.

We have verified the technology, and space requirements, but have not yet integrated the final power supply with the GEECON.

7.4 Daughter board

In order to interface our GEECONwith the DT-VGT, we have developed a DT-VGT specific daughter board that interfaces the FPGA mother board of the GEECON with the sensor and actuator systems of the DT-VGT prototype, as shown in figure 7.3.

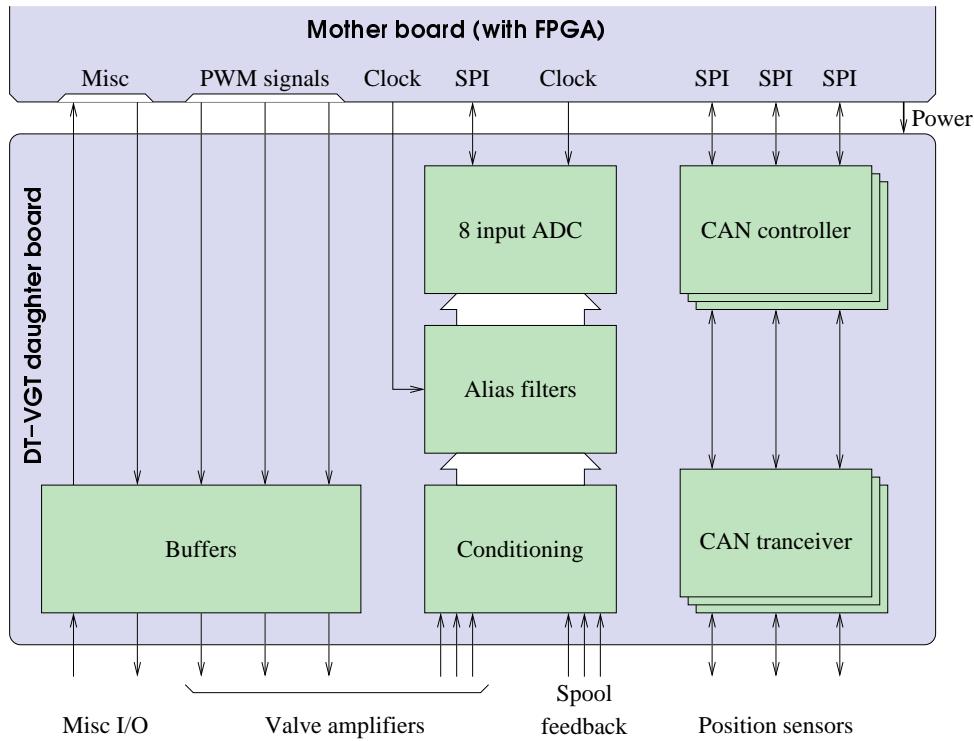


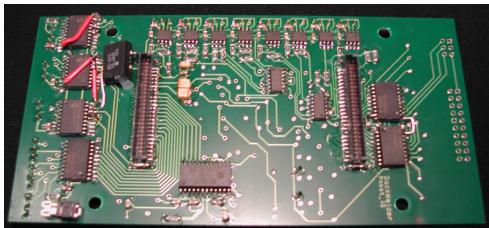
Figure 7.3: Block diagram of the DT-VGT daughter board

The detailed circuit and PCB design have been performed by Benny Jørgensen and Danny Kyrping, and the documentation can be found in [Kyrping-B]. The most interesting features of the design are described below.

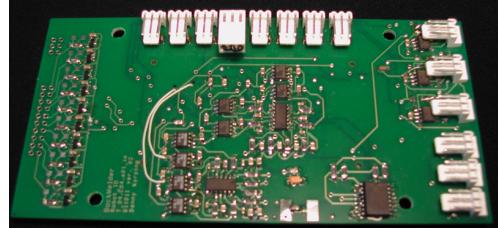
- The digital PWM signals for the valve amplifiers are provided by simple buffers.
- The analog voltage signals from the valve amplifiers and the optional spool position sensors are scaled and offset to match the ADC input range, and filtered with 4. order switched capacitor Butterowrth low pass filters, before the signals are fed to the ADC.
- Providing the switched capacitor filters with a clock signal from the mother board, allow us to adapt the cut off frequency of the filters to the sample rate we chose to use.
- We use the 14 bit, 8 channel, 285kSPS A/D converter AD7856 [AD7856], that communicates through a SPI interface. This saves both board space and FPGA signals, at the expense of increased FPGA configuration complexity.
- The CAN-bus interface for the Temposonic position sensors are implemented using PCA82C250 transceivers and MCP2510 stand alone CAN controllers with SPI interfaces [MCP2510].
- As the interface between transceiver and controller are purely digital, we could reduce the complexity of the daughter board, by implementing the CAN controllers in the FPGA, either by specifying a CAN controller in VHDL, or by buying a CAN controller *core*

for integration with our FPGA specification. As we could afford the board space for the stand alone controllers, we prefer this solution as it is far simpler than implementing or integrating a CAN bus controller in our FPGA specification.

- To be prepared for future introduction of monitoring, safety and other equipment, we have set a number of buffered digital inputs and outputs aside.



(a) Motherboard side



(b) I/O connector side

Figure 7.4: The DT-VGT daughter board

As indicated by the photos, the necessary I/O circuitry for the DT-VGT interface can easily be accommodated by a PCB of the same size as the mother board. It is also clear from the photos that a few design mistakes have to be corrected in future versions.

7.5 FPGA configuration

In order to integrate the DT-VGT through the daughter board, 3 different VHDL I/O modules have been designed, and integrated into the framework of the generic FPGA based I/O framework described earlier. They are:

- PWM generator
- ADC interface
- Temposonic interface

PWM generator

Generates a fixed frequency PWM signal, consistent with the specifications of the valve amplifiers. The pulse width is controlled by the CPU through a memory register (signed int) in the FPGA. This module is instantiated 3 times in order to provide PWM signals for all 3 valve amplifiers.

When we replace the current *stand alone* valve amplifiers designed by Jakob Lindeløv, with more compact, stripped down, versions we will merge the VHDL specification of the simple PWM generator with the VHDL used to implement switch patterns and current limit in the CPLD of the valve amplifiers.

ADC interface

Handles the SPI communication and other relevant control of the AD7856 ADC on the daughter board. This module features a state machine that initializes the ADC upon system RESET, and performs cyclic (round robin) measurements from each of the 8 ADC channel thereafter. The measurements are made available for the CPU through an internal dual ported RAM, making the whole ADC implementation transparent for the CPU, which can simply read the latest value of any ADC channel from a memory mapped I/O register at will.

Temposonic interface

The basis of this module is a dedicated SPI interface, that supports a subset of the SPI commands used by the MCP2510 CAN controllers on the daughter board. This SPI- or rather MCP2510 interface is accessible from the CPU, and can be used to control and communicate with the MCP2510. On top of the MCP2510 interface, we have implemented a state machine that uses the MCP2510 interface to receive and interpret CAN messages from the Temposonic position sensors of the VGT. Each time a message arrives, the state machine extracts position, velocity and status information, storing each value in an internal memory register accessible for the CPU and optionally generating an interrupt. The MCP2510 and the Temposonic sensors must be properly initialized by the CPU, using the low level MCP2510 interface, before the high level Temposonic interface is engaged.

The development of this component, through several states, is described in more detail in [Sørensen-G]

Implementation

The FPGA configuration was originally implemented for the XC40xx FPGA of our 32-bit prototype FPGA I/O board, but have recently been ported and verified for the *Spartan II* motherboard by Jack Knudsen. As the *XC40xx* implementation have been used during the work reported here, we refer to the original *XC40xx* implementation [Sørensen-F], where the designs can be found in the files:

- VHDL/top_level.vhd
- VHDL/simple_pwm.vhd
- VHDL/adconv.vhd

- VHDL/can.vhd
- VHDL/id_rom.vhd
- VHDL/irqctrl.vhd

In addition to the 3 different I/O modules, the FPGA configuration also features an identity ROM, used to identify the FPGA configuration and version to the host computer, and an interrupt controller used to handle the optional interrupts from the I/O modules.

7.6 Software

Software development for the GEECONs was originally intended for parallel projects, which have either failed to complete or to initialize. The software we have developed for the GEECONs is merely intended to demonstrate the system components and inspire further development, not to form the basis of a full scale distributed control system.

Although we have chosen a software architecture with multiple parallel tasks, we do neither use nor recommend using an operating system, as the required architecture is both simple and static. The tasks requiring real-time performance are implemented as interrupt service routines, triggered by periodic interrupts, generated by I/O circuitry or timers. Non real-time tasks, like initialisation and overall system control are implemented as a normal program executing in the background.

In order to handle blocks of data, like configuration data, joint streams, sensor and event logs etc. we have implemented support for named data modules. We have implemented a small library, that allow us to create, locate and operate on named data modules. We have also created a set of Unix-tools that allow us to transform various files into named modules that can be linked with our programs, or transferred to the GEECON in other manners.

We refer to the source code, including the make file, in the implementation example [Sørensen-D] for documentation of our software.

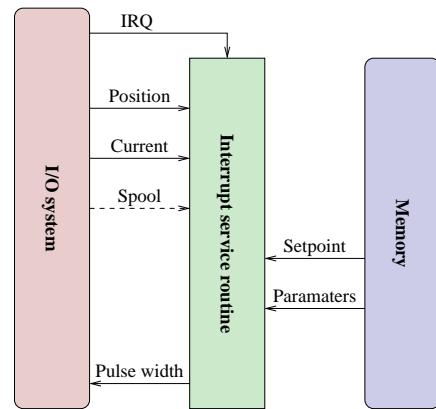
Control layer implementation

During initialization, we set up each of the 3 Temposonic position sensors to report the actuator position at their maximum rate of 2kHz. Each time a position is reported, our FPGA based I/O system will place the position reading in a readable hardware register and generate an interrupt.

As the 3 sensors are not synchronized, they will report at slightly different frequencies, and we must handle their feedback individually in order to avoid inter modulation effects.

In effect, each actuator system is serviced by an individual interrupt service routine, that:

- Read the reported position
- Read other relevant input data
- Read the current actuator set point
- Evaluates a control law in order to calculate a suitable output.



As this procedure is repeated approximately 6000 times per second, it becomes relevant to optimize its operation. We do this primarily by:

- Avoiding any unit conversion, by ensuring that all parameters are kept in units compatible with the input output system.
- Ensuring that all parameters used are stored in internal DSP registers with fast access.
- Keeping computational efficiency in mind when devising and implementing control laws.

We have been contend to implement these interrupt service routines in C using the Texas Instruments C compiler accompanying the CPU modules. We have not yet gained sufficient experience with low level DSP programming to asses the compilers ability to optimize the code, and have made no attempt to *hand optimize* it ourselves.

An example of the interrupt service routine can be found in [Sørensen-D] in the file `isr.c`.

In order to allow both efficiency and clarity, the on line parameters used by the interrupt service routines, are generated from configuration data modules during system initialization. The configuration data modules contains all relevant information about I/O scaling as well as parameters for the relevant control law in SI units. We operate with configuration modules for the AD converter, and the 3 individual actuator systems. While the AD configuration module contain calibrated values for offset [bits] and gain [bits/V] for the individual AD channels, the configuration modules for the individual actuator systems contain information about e.g.

- Settings for the Temposonic position sensors.
- Measured valve properties (resistance and inductance)
- Control parameters for the valve
- Valve amplifiers and PWM generator data e.g. supply voltage, output range and resolution.
- Actuator length, position sensor offset, sensor resolution and parameters for position control.

Examples of configuration modules can be seen in [Sørensen-D] in `adc_module.src` and `actuator_mod1.src`

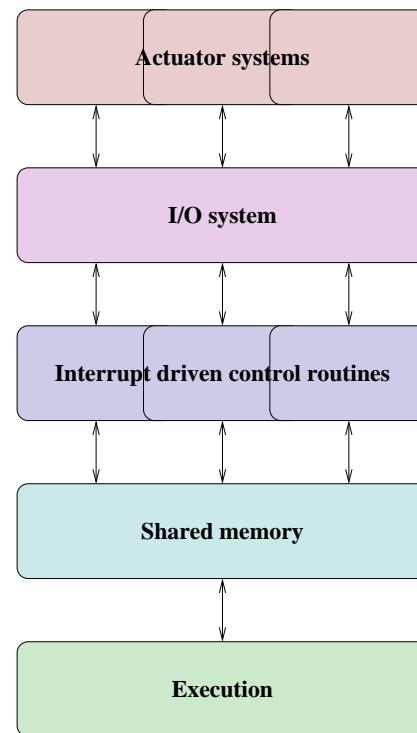
Execution layer implementation

When the system has been initialized, the interrupt driven I/O and control system described above forms a virtually autonomous position control system, ensuring that the position of each actuator follows a given set point, which is continuously read from a shared memory register. This constitutes a very clear example of a control layer implementation.

During initialisation, the set point of each actuator is set equal to the actual position of the actuator, to avoid discontinuities in the input to the control system. When the system is initialized, the actuators can be moved by the execution layer implementation, by dynamically updating the set points for each control system.

As the current DT-VGT's are neither supposed to be very accurate, nor very fast, the joint streams used to specify their motion, have sample frequencies well below 100Hz, and we have used 25Hz or 50Hz throughout our work with the DT-VGT. The low bandwidth of the hydraulic actuators and the fact that we run the robot off-line have allowed us to use 0. order extrapolation for our previous demonstrations and tests.

As we have not yet had opportunity to integrate the network components of the GEECON into the system, our execution layer software is simply an interrupt service routine driven by a timer, that will read a joint stream from memory. The timer is set at a the sample frequency of the joint stream, in order to avoid resampling. Our experience with the DT-VGT have shown that a sample frequency of 50Hz is sufficient to suppress alias noise to an acceptable level. As this frequency is substantially lower than the frequency of the control loop, we do not need to take special precautions against inter modulation of the execution and control loops. As a set point is stored in a single register, changing it is an atomic operation that requires no protection from context shifts in the program execution.



Future work

Evidently, the next development step is to integrate the network interface with the execution layer implementation. This work is currently being planned in cooperation with our project partners,

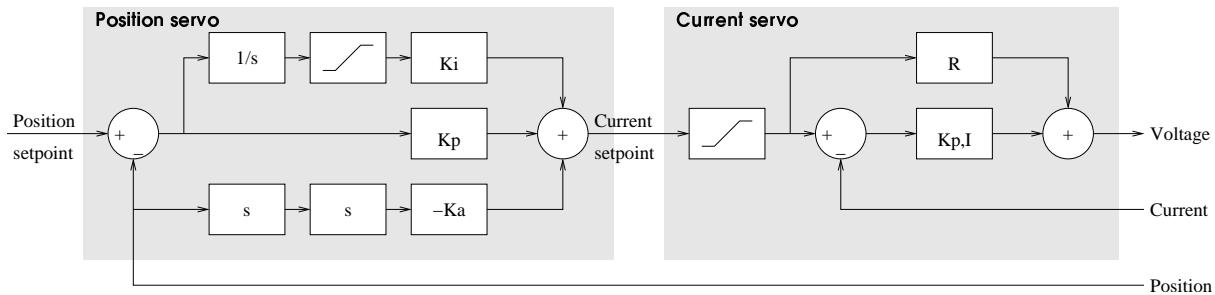


Figure 7.5: Control system for hydraulic actuator

in order to demonstrate an integrated system with two aggregated DT-VGT modules during mid 2003.

We expect that the experience gained from this integration and demonstration will inspire software development that will support the components needed to achieve our vision of *plug and play* reconfigurable modular robots.

7.7 Hydraulic control

As with the software, the control system for the DT-VGT actuators were supposed to be developed in a parallel project that did not become a reality. In order to demonstrate our technology, we have implemented a simple control algorithm that works satisfactory for simple demonstrations.

The master thesis of Carsten Olsen [Olsen00] have proved very helpful throughout this task, as Carsten Olsen studied hydraulic control of a previous VGT prototype, that have some factors in common with the one used by us. From a control perspective, the differences between our system and the previous prototype lies in:

- The static and dynamic load of the actuators are quite different, due to changes in the geometry of the DT-VGT.
- The valve amplifiers have been exchanged.
- We have access to current feedback from the valve amplifiers
- The sample rate of the position sensors have been increased from 1 to 2kHz.

We have implemented the position servo recommended by [Olsen00], which is simply a P-I controller with acceleration feedback to suppress vibrations. The structure is indicated in figure 7.5.

In [Olsen00], a conventional P or P-I controller proved to be unstable, and it was speculated that the instabilities were due to resonances formed by the compressibility of the hydraulic fluid. It

was possible for [Olsen00] to stabilize the system by differentiating the position twice in order to estimate the acceleration, and use it to dampen vibrations. We have duplicated the position servo of [Olsen00], but was only able to reproduce the reported instabilities in experiments where we replaced our own valve amplifiers with the ones used in [Olsen00].

We have not dedicated resources to a detailed analysis of the previous valve amplifiers, but as their linearity and dynamic properties were never challenged in [Olsen00], it seems likely that the source of the reported instabilities should be found in the amplifiers, rather than the hydraulic components.

As the source of the reported instabilities have not been finally identified, they might return under changed conditions, and we have merely disabled the acceleration feedback, rather than removed it.

We have not yet demonstrated the DT-VGT for tasks requiring high precision, and no attempts have yet been made to optimize the controller, or to document its performance. Simple observations during tests and demonstrations indicate that the controller must be improved if the DT-VGT's are to be used for accurate work like e.g. arc welding.

Modelling

A logical first step towards improving the controller, is to obtain accurate understanding and models for the hydraulic actuator system. Accurate models can be used to simulate the system when designing and testing controllers, and if they are computational effective, they can be used to implement model based feed forward controllers.

The models developed in [Olsen00], helped the design of the current controller structure, and the estimation of control parameters for the previous DT-VGT prototype, but they were not accurate enough to support model based feed forward.

During this project we have proposed a modelling method that produces simple, yet very accurate models of a single hydraulic actuator system. The method represent a combination of mechanical analysis, fuzzy modelling, and linear signal analysis. The model parameters are obtained from analysis of a series of different open loop experiments, but unlike many popular *system identification* methods, our method relates to known physical properties of the hydraulic system, which are merely parameterised.

As the experiments and analysis required have not yet been automated, they are quite cumbersome, and we have only modelled a single actuator system so far. Verification of the model obtained, using independent open loop experiments, indicate that we can model the dynamic valve-current to actuator-speed relationship with an accuracy better than 10% RMS, for motions involving moderate speed and acceleration.

Our modelling method and our results are described in [Sørensen-E], and selected simulation results from that document is pasted in figure 7.6 below.

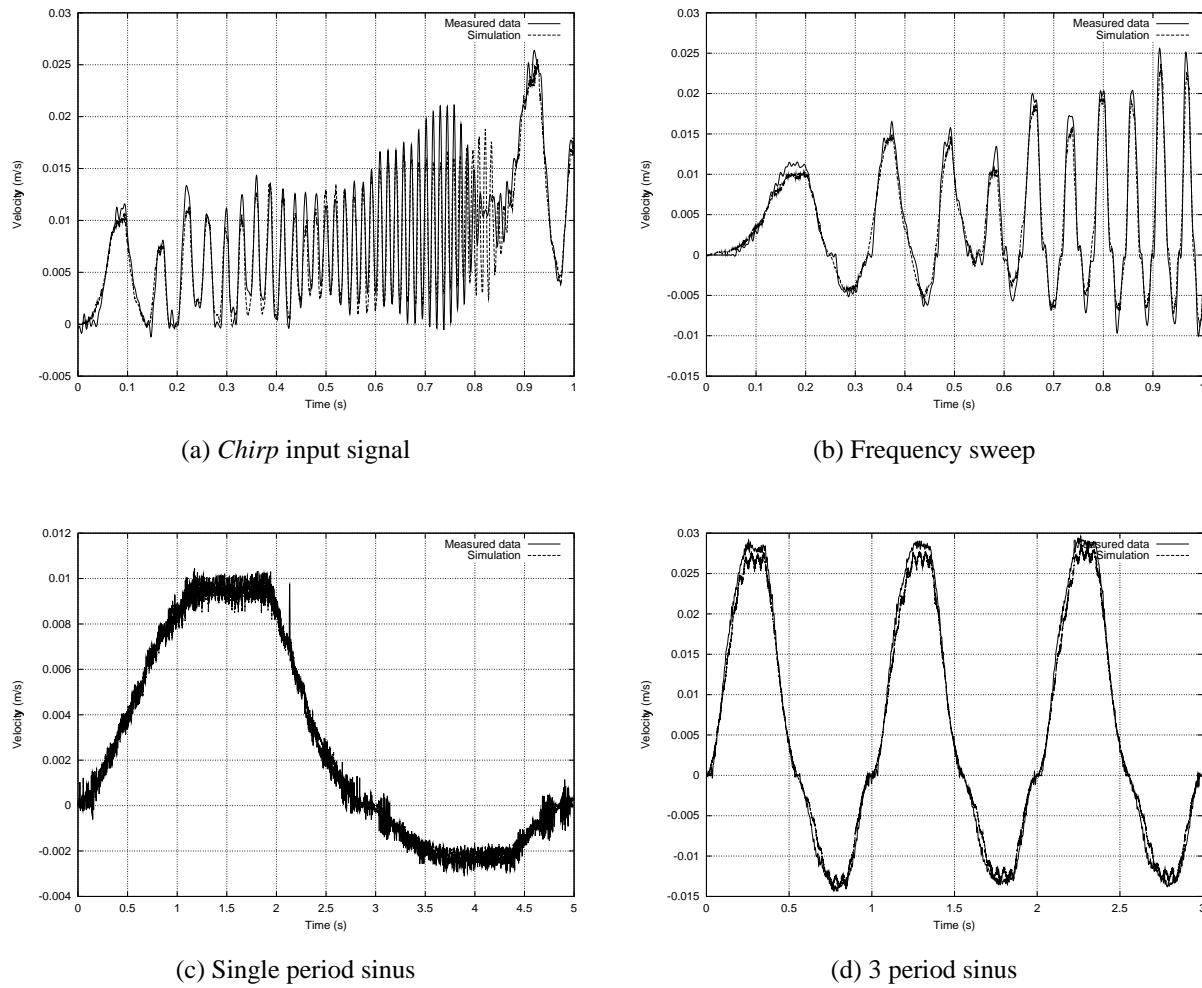


Figure 7.6: Comparison of simulated and real responses for open loop experiments

7.8 Mechanical integration

The mother board with CPU module and daughter board have been mounted in a rectangular enclosure, that fits inside the cylindrical body of the DT-VGT — see figure 7.7. In addition to the control computer we must mount a set of miniaturized valve amplifiers and a 300VDC to 48VDC converter inside the enclosure, before we can complete the mechanical integration, mounting the enclosure inside the VGT and shortening the connecting cables. When the control node is physically integrated, it becomes hard to approach physically, and we are not very eager to complete this step until we are completely convinced we do no longer need frequent physical access to the components inside the enclosure.

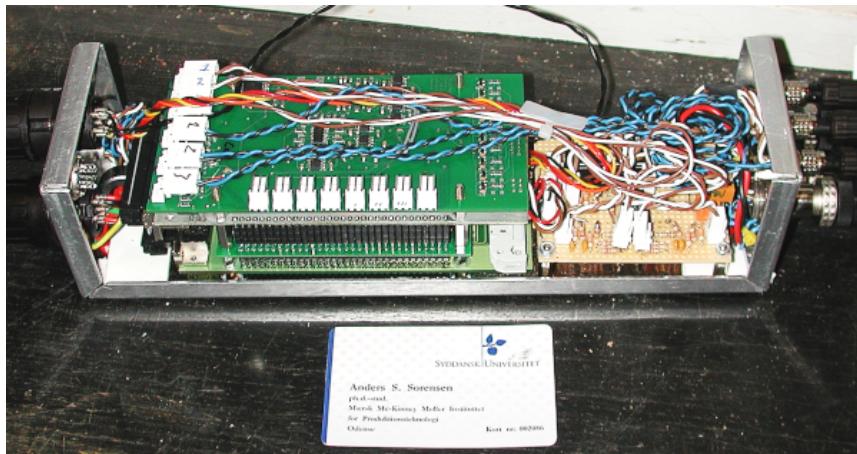


Figure 7.7: GEECON prototype mounted in enclosure

7.9 Experience

During the course of this project, we have been controlling the DT-VGT with the GEECON for hundreds of hours. Although minor implementation glitches have been discovered, there is no doubt that the GEECON is fully capable of controlling the DT-VGT in the intended fashion. Although the DT-VGT as well as the controller have been in a constant state of change, the GEECON have worked reliably throughout our work with the DT-VGT.

It has been a positive experience to be able to alter the I/O system as easily as we are used to altering software. It is very efficient to be able to adapt the two aspects of interfacing to each other, rather than the normal way of devising elaborate software to catch up with non optimal hardware.

7.10 Conclusion

Our generic embedded control node (GEECON) have successfully been integrated with the DT-VGT with respect to I/O and control. The final mechanical integration of the two awaits the completion of a few tasks, that are not expected to cause problems.

The relative ease with which we have implemented a compact and elegant I/O interface between the GEECON and the robot, is a good demonstration of the flexibility and power of our generic I/O system.

Our ability to design, manufacture and integrate a powerful and flexible control platform with a mechanism as special as the DT-VGT, is an excellent milestone on our quest for modular control.

It is unfortunate that it has not been possible to initialize or sustain the parallel projects that should have boosted our efforts with respect to low level software and control, but we have

managed to implement minimalistic solutions that allow us to demonstrate our technology, even if there are clearly room for improvements on both accounts.

Chapter 8

PA-10 interface

Although we have chosen the DT-VGT robotic modules as our primary demonstrator for this project, we find the inclusion of the PA-10 robot to be significant for the following reasons:

- It demonstrates that our GEECONs can be used with different mechanisms
- While the DT-VGT modules have many advantages, they only have 3 degrees of freedom, and a limited work space. Adding the 7 degree of freedom PA-10 robot to a DT-VGT give a very impressive aggregated robot that combines the advantages of parallel and sequential kinematic mechanisms.



8.1 Description

A PA-10 robot system consists of a 7 degree of freedom robot arm, connected to a separate servo controller via 2 40mm cables.

The arm

The arm consist of a base with a rotational joint, and 3 arm segments, each with a tilt and rotational joint. The arm is ideal for accessing narrow spaces, as it is very slender, symmetrical, and has a completely symmetrical work space. The 7 degrees of freedom ensures a redundant forward kinematic which enables the arm to position its tool in multiple ways, potentially avoiding obstacles in the environment.

The arm is powered by 7 individual AC servo motors embedded in the arm, and coupled to the mechanics with harmonic drives.

Position feed back is obtained by two resolvers for each motor. One resolver monitors the angle of the motor shaft, while the other resolver monitors the angle of the joint in question. As the harmonic drives are virtually free of backlash, this setup enables the joint positions to be measured with tremendous accuracy.

The cables

One cable transfers power to the 7 motors, while the other carries the analog signals for the 14 resolvers.

The servo controller

The servo controller features a 1kW power supply for the servo drivers, 7 AC servo drivers, 14 resolver interfaces, and 4 digital signal processors to control the works.

The controller implement an individual force (current) control loop for each motor, nested inside an optional velocity control loop. All communication of commands, status, parameters, set points and feed back is conducted through a very simple protocol on a 5Mbps ARC-net network.

The ARC-net used by the PA-10 controller use an optical fiber media, that can only connect to one node besides the PA-10 controller.

Communication

When the robot is initialised and activated, a vector with set points for each actuator — either velocity or force — is transmitted to the controller with a sample period of no more than 30ms. If this interval is exceeded, the controller disables the robot and issues an error signal. Each vector is transmitted in a single ARC-net package.

In response to each package with set points, the controller transmits an ARC-net package with position feed back and status for the actuators.

8.2 Observations

Robot arms like the PA-10, does not fit completely with our vision for a modular system, as the servo controller is placed physically distant from the arm itself. As the most logical placement for one of our generic embedded control nodes (GEECONs) will be inside, or close to the native

controller, the physical placement of the arm, will not be reflected by the configuration of the network connecting the GEECONs.

Unless we want to strap a GEECON to the arm and run an optical fiber connection down to the native controller, the central controller software must be designed to allow some sort of intervention to an automatic configuration based on the network topology.

8.3 ARC-net interface

All that is necessary to interface our GEECON to the PA-10 is an ARC-net interface with appropriate optical transceivers. The task of designing such an interface was out sourced to Mads Lundstrøm, as part of a bachelor thesis project [Lundstrøm01]

Initially Mads used our generic I/O system to interface a COM20022 ARC-net controller IC to the GEECON. As it turned out that the COM20022 could be connected directly to the Microline bus of the DSP, with the aid of a few logical gates, we decided to implement the ARC-net interface as a stand alone Microline module, as no benefits could be gained from integrating it with the generic I/O system.

8.4 Software

Mads Lundstrøm proceeded to write the PA-10 control layer software for the GEECONs. Like the software written for the DT-VGT modules, the PA-10 software features an independent control and execution loop, exchanging set points through shared memory. Both the execution and control loop is implemented as interrupt service routines, driven by timers and the ARC-net interface.

Mads Lundstrøm worked in parallel with Michel Bøllingtoft. In Michel's bachelor thesis project[Nielsen02], a PC running a real-time version of Linux was equipped with an ARC-net interface, and a suitable ARC-net device driver was developed, in order to allow real-time Linux applications to communicate with the GEECONs.

As a demonstration of their respective projects, Mads and Michel successfully made a simple implementation of execution layer software for the GEECON as well as the central Linux PC, that allowed the PA-10 to be controlled on-line from real-time applications on the central Linux PC.

8.5 Experience

Mads and Michaels demonstration of on-line control of a robot through the backbone network is significant, as it demonstrates the GEECONs ability to control a mechanical unit in accordance with a joint stream being delivered on-line from a central controller.

While the demonstration system developed by Mads and Michael is too simple to be used for more complex aggregated robots, their experience is valuable to the further development and system integration.

The development of the PA-10 interface have been relatively uneventful, as our prior experience with ARC-net as well as the PA-10 have helped to avoid major problems. Apart from the mechanical design of the ARC-net interface, we have been able to use Mads Lundstrøms work unaltered, for several successful demonstrations involving the PA-10 robot.

8.6 Conclusion

The PA-10 robot can be reliably controlled by the interface and software developed for the generic embedded control nodes (GEECONs), and we have successfully used the PA-10 with it's GEECON for several demonstrations.

In addition to demonstrating that our GEECON can control a PA-10 robot, it has been demonstrated that they can control a host mechanism in accordance with a joint stream received on-line through their integrated ARC-net interface. The software involved in this demonstration was developed as part of two bachelor thesis projects, which have provided valuable experience and ideas for further systems integration.

8.7 Future work

The execution layer software developed by Mads Lundstrøm and Michael Bøllingtoft must be rewritten in accordance with the protocols to be defined for general integration between the GEECONs and the central controller.

Chapter 9

System integration

During this project, we have successfully integrated our generic embedded control nodes (GEECONs) with a DT-VGT robotic module and a PA-10 industrial robot arm. This *local* system integration is described in chapter 7 and 8. In addition to these applications, the GEECONs have been integrated with various other equipment, external to this project.

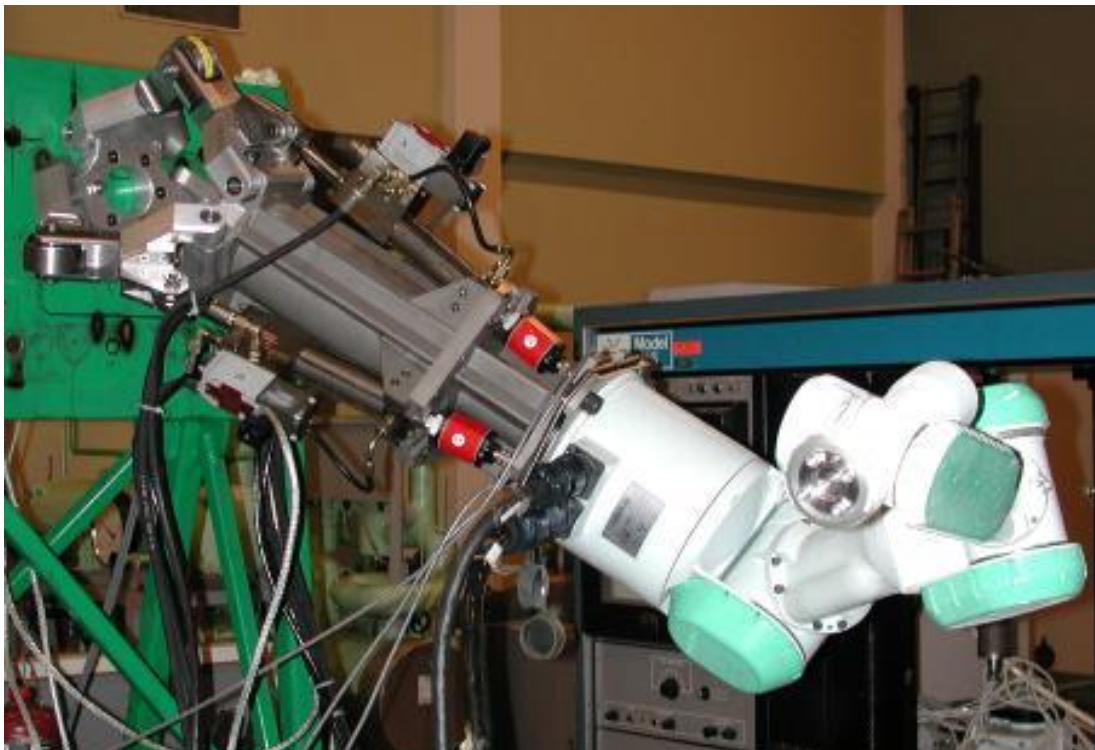


Figure 9.1: Initial test of PA-10 robot aggregated with a DT-VGT module

In order to demonstrate the *global* systems integration involved in aggregating robot modules

controlled by GEECONs, we have demonstrated that the DT-VGT and PA-10 mechanisms can be aggregated and controlled as a single mechanism, by attaching the PA-10 to the DT-VGT and using the resulting 10 degree of freedom mechanism to spray paint the pan of a wheel barrow. At this point we have also defined the overall architecture to be used to integrate our project with the EU project *DockWelder*, where 2 DT-VGT modules and a customized *revolute joint* mechanism will be aggregated with a 6 degree of freedom robot arm.

The wheel barrow demonstration, and the system being designed for the DockWelder project represent the first steps toward integrating our technology into a full scale robotic system, and are described in more detail below.

9.1 Central controller

The GEECONs have successfully been used to control the DT-VGT and the PA-10 as individual stand alone systems, but in order to use the DT-VGT and PA-10 as a single aggregated robot, we need to integrate the GEECONs with a central control node that can control them in a coordinated fashion — see section 3.3.

During this project, we have been planning to cooperate with Odense Steel Shipyard, to integrate our distributed controller architecture with their *Open modular controller* — see section 2.5, by specifying and implementing a simple network protocol, which would have allowed us to come fairly close to our original vision of a distributed control system.

Unfortunately, Odense Steel Shipyard have recently decided to abandon further development and use of the *Open modular controller*, leaving us no option for integrating our distributed architecture with an advanced industrial controller within this project.

9.2 Painting a wheel barrow

To demonstrate the system without Odense Steel Shipyard and their Open modular controller, we have cooperated with a number of other local partners that had an interest in demonstrating an industrial process performed by an aggregated robot. Instead of using an *on line* central controller, we have used some of the essential central components to obtain an *off-line* specification of the robot motion and process.

As the demonstration was performed in cooperation with the company Meganic Aps, the *Smart painter* research group and the motion planning research groups from the Mærsk Institute, it was in fact several demonstrations in one. We demonstrated that:

- The DT-VGT technology is useful for industrial applications.

- The local research group dedicated to spray painting (Smartpainter) can generate good tool/nozzle paths for concave objects.
- The local research group dedicated to motion planning can convert tool paths to joint paths for redundant robots containing parallel kinematic elements.
- It is possible to use our GEECONs to control an aggregated robot in accordance with a given joint path.

System setup

Although the components used for this demonstration are quite loosely coupled and used off line, they constitute some of the major components of a central controller. The same or similar components will be used in later on-line versions of a central controller, and have indeed been used earlier in the Open modular controller before it's termination.

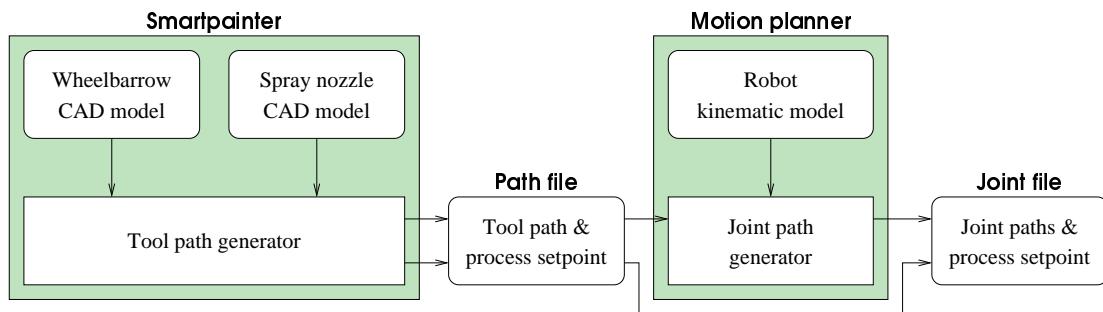


Figure 9.2: Overview of planning system for paint demonstration

Figure 9.2 shows the components used to obtain the joint paths for the aggregated robot. First, models of the wheel barrow and paint nozzle are used to specify a nozzle path, combined with a set point for the paint process — paint on/off in this case. The combined path and process specification is stored in a file for later processing.

The motion planner use a kinematic model of the aggregated robot, to transform the tool-path specification into path specifications for the 10 individual joints of the aggregated robot. The process specification is encoded into the joint stream as an 11. joint, and the entire specification is stored in a *joint file* for later processing.

Figure 9.3 shows the system used to execute the paths generated by the planning system.

Two GEECONs are used for the aggregated robot. We equipped the one controlling the PA-10 with a suitable digital output to turn the spray paint on and off. The joint file is first split into individual files for the two control nodes. The files are converted to data modules, which are transferred to the two GEECONs.

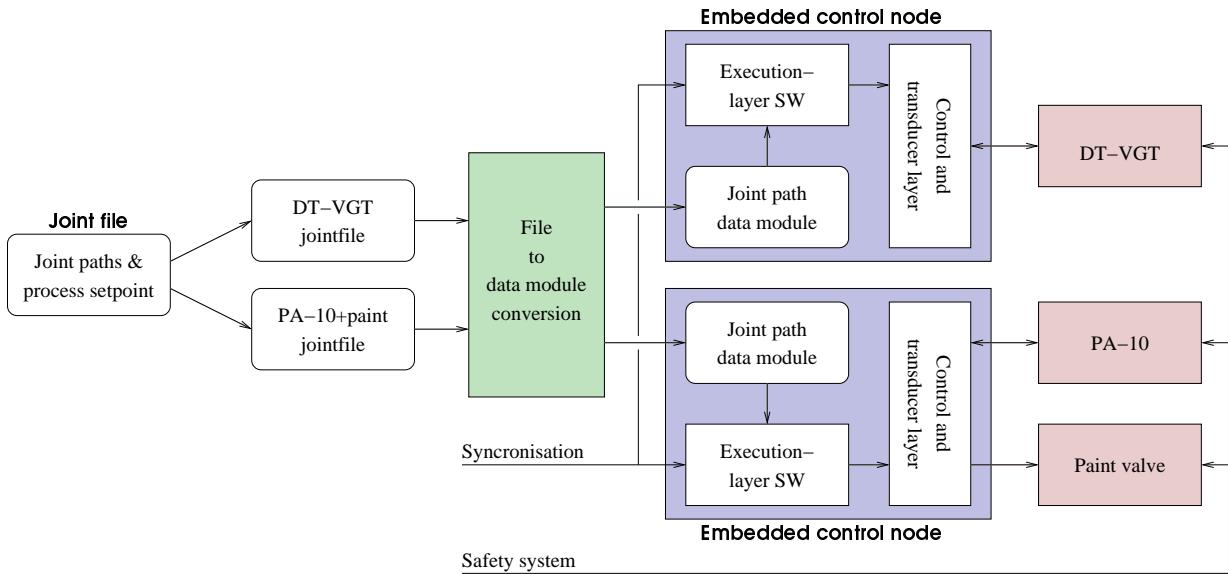


Figure 9.3: Overview of execution system for paint demonstration

The GEECONs are synchronized, simply by issuing the command to execute the joint paths simultaneously.

We have not yet integrated safety systems into the GEECONs, and rely on a crude but effective external safety system that removes the hydraulic and electrical power supply from the mechanics in case of an emergency stop.

Experience

The only work needed to integrate the PA-10 robot and the DT-VGT with each other, was to attach them mechanically, calibrate/measure the properties of the mechanical connection through external means, and connecting their GEECONs to the same *start button*. As the GEECONs of this particular demonstration were controlled with commands through their RS-232 ports, the *start button* took the form of a PC with a dual RS-232 interface.

When network communication is included in the execution layer software, the system integration will be even simpler, as the RS-232 connections will simply be replaced by a common network cable.

The wheel barrow demonstration have been performed several times, with different tool paths. Although the setup is rather simple, it has provided all partners with a reliable and convincing demonstration of our respective technologies. Except from the complexity associated with using the primitive setup, the only flaw we encountered was a tendency for the paint nozzle to clog during painting, a problem that has to do with paint-chemistry and fluid mechanics which are clearly outside the scope of our project.

9.3 The DockWelder system

When we implement the setup for the DockWelder demonstration, we will have re implemented the current execution layer software to support on line joint streams delivered over the common communication network of the nodes. We imagine the overall structure of the DockWelder system to resemble figure 9.5

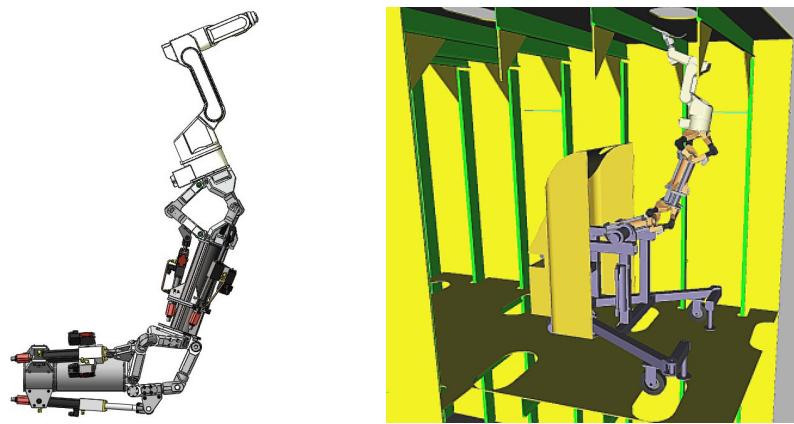


Figure 9.4: Sketches of mechanics for DockWelder

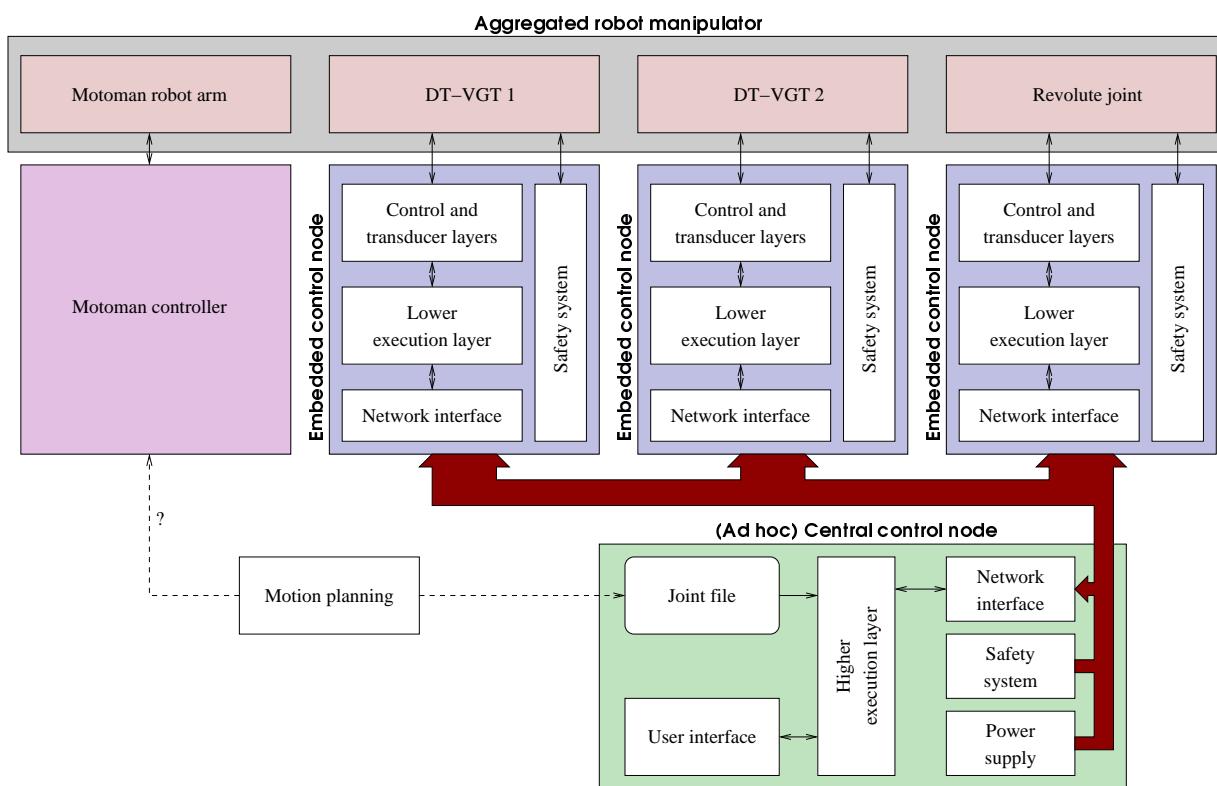


Figure 9.5: Overview of execution system for DockWelder demonstration

One of our foreign DockWelder partners are in charge of controlling the Motoman robot arm, and it is not to be included in the modular control system at this time.

The system structure is somewhat similar to the system used for the paint demonstration de-

scribed above. The primary difference is that joint streams will be transmitted on line, rather than stored in the GEECONs, and that we wish to integrate the safety system with the GEECONs.

The DockWelder system will also feature manual control of the entire aggregated robot, except the Motoman arm, from the central controller.

The system will not feature on line motion planning, as the joint files will still be generated in advance, for off line execution at a later point.

In the initial system, the revolute joint and the two DT-VGT's will be used exclusively as a placer mechanism for the Motoman arm, and the Motoman arm will not move simultaneous with the rest of the aggregated robot.

9.4 Conclusion

We have successfully integrated two different mechanical modules, controlled by our generic embedded control nodes (GEECONs), into a 10 degree of freedom aggregated robot manipulator, and demonstrated it's ability to perform a typical industrial process.

The demonstration have shown that integration of mechanical modules controlled by our GEECONs is primarily a matter of mechanical connection, as no signals apart from the GEECON communication need to be connected to a central controller.

The wheel barrow demonstration and the proposed DockWelder demonstration system represent the two first steps toward utilizing the potential of mechanical modules controlled by GEECONs.

9.5 Future work

The majority of work within system integration clearly lie in the development of the central control node. This development is currently only supported by a single Ph.D. project, which is hardly enough to ensure that our modular control architecture evolves into a practical useful system.

We suggest that cooperation with external partners is initiated, in order to investigate the possibility of integrating our distributed modular controllers with existing controller technologies such as The GENERIS controller described in section 2.5. At the same time it would not be a bad idea to strengthen the local research and development in high level controller technology.

Chapter 10

Experience and discussion

During this project we have worked with, developed and integrated several interesting sub projects and technologies. In this chapter we will summarize and comment on our main experiences.

10.1 Generic controller modules

During this project, we have outlined, designed and implemented a generic controller platform that have allowed us to experiment with and demonstrate our idea of a generic embedded control node (GEECON).

We have successfully placed an experimental DT-VGT parallel kinematic manipulator module as well as a conventional industrial robot under the control of our generic controller platform, and successfully demonstrated that the two mechanisms can be aggregated and controlled as a single manipulator.

In addition to our own experiments, our generic controller platform have been used in a number of unrelated projects, where it has been interfaced to:

- **An experimental ultrasonic scanner**, where the controller module recorded and processed analog data at a sample rate of 60MHz, passing the processed data along at a rate of 5MHz.
- **A prototype Double Octahedron variable geometry truss**, where the controller module interfaced directly to electrical motors, and performed the same type of position control used for the DT-VGT and Pa-10 robot in this project.
- **A mobile robot**, where the controller interfaced directly to electrical motors and various external sensors.

The experience gained from all these projects, along with a number of minor experiments, all infer that our generic controller technology is sufficiently flexible to interface to any sensor or

actuator technology in our experience. This is supported by our demonstration of interfaces for a wide range of analog and digital signals, as well as parallel and serial communication systems.

Our main concern with respect to the GEECON flexibility is that the number of I/O signals is limited, by the number of pins on the FPGA used on the GEECON motherboard¹. This might either limit the complexity of mechanical host modules, or force us to expand the I/O system of the GEECON. The I/O system can be expanded by adding additional (FPGA based) I/O boards to the Microline bus, or by redesigning the motherboard to accept an FPGA with more I/O pins. As FPGA's exist with over a thousand I/O connections, we conclude that this limitation in GEECON flexibility is caused by our specific design, rather than the properties of the underlying technology.

In section 3.6 we recommended to “aim for as high I/O bandwidth as practical convenient” based on examples of interface layer components with bandwidths well into the MHz range. We also estimated a minimum requirement of 4kHz per signal. The I/O performance demonstrated by the GEECON in some of the above applications is well in compliance with both requirement and recommendation. In fact, we have never experienced a mechatronic system requiring I/O performance in excess of our system, nor do we expect to find one.

In order to allow our generic controller to be integrated with the DT-VGT and other systems that impose strict demands on the controller size, we have designed a very compact controller platform, that can be accommodated by a rectangular space down to approximately $14 \times 7 \times 5$ cm. It has taken several design iterations to arrive at such a compact design, and our primary motivation to go this far is our commitment to deliver practical prototypes for the DT-VGT mechanism. In our experience, the current GEECON is small enough to be embedded within most industrial mechanisms or their native controllers. It is not possible to achieve further significant size reduction with the technology used in this project. A more compact GEECON may however be realized by abandoning the commercial CPU platform in favor of an entirely customized design.

In chapter 3, we estimated that a performance of 1 million operations per second is sufficient to implement simple control laws, for a GEECON handling up to 10 actuators. Our GEECON utilizes a commercially available DSP module with a performance of 40MFLOP/20MIPS, giving us more than sufficient processing power to control a DT-VGT or a PA-10, with our current linear control laws. As compatible DSP modules with performance up to 900MFLOP are available, we assume that our controller platform will be able to support the processing demands of most industrial mechanisms, even if they demand more advanced control algorithms. The CPU performance rating is theoretical, and the practical performance relies on the programmers ability to utilize the CPU architecture efficiently. The control laws and corresponding software of our demonstration systems are exceedingly simple, so the efficiency of the implementation has not been an issue. When more complex control laws and software is developed, care must be taken to ensure a sufficiently efficient implementation.

Although we have been using the controller modules extensively for experiments and demonstrations, we have experienced no technical problems beyond simple assembly and programming

¹In the current design, 58 FPGA pins out of 208 are allocated to GEECON I/O

mistakes. As our current prototypes are only designed for laboratory use, it is irrelevant to test them in accordance with industrial standards, but we are confident that our technology will pass the relevant tests once the connectors along with the environmental and electromagnetic sealing have been upgraded for industrial use.

10.2 Aggregated modules

Even though we were prevented from integrating our distributed controllers with the central controller technology developed by Odense Steel Shipyard, we have experimented with, and demonstrated the ability to aggregate two very different mechanisms controlled by our GEECONs, and use the aggregated mechanism as an integrated system that can perform industrial processes. As described in chapter 9, this demonstration went well and we are looking forward to extending the system with more mechanical modules and a more sophisticated central controller.

During our interaction with the DockWelder project, an ad-hoc central controller will be implemented, in order to control the mechanical modules on-line. As the primary software components of a central controller are still available in the local robotics community, we are confident that the ad-hoc controller being developed for DockWelder can evolve to utilize the full benefits of our distributed low level controller architecture, including automatic or semi-automatic configuration based on the physical topology of the modular mechanism, and thus the network of distributed nodes.

10.3 The DT-VGT

During our experiments we have gained substantial experience with the DT-VGT and the technology used to implement it. Much of the experience is irrelevant to this project, as it regards the mechanical design of the prototype, but it has been welcomed by the company Meganic, which have incorporated much of the experience gained in their design for the next prototype.

It has been very interesting to work with hydraulic actuators, and although they are highly impractical compared to electric motors, our experience with the technology have been quite positive. We were able to suggest a promising method for modelling actuator response, which can be used to devise control algorithms with better performance than the simple linear regulator used for present demonstrations.

We are impressed with the performance of the DT-VGT, which seems to be a very promising technology for tasks requiring slender high strength mechanics, with high dexterity regarding angular displacement. At the same time the DT-VGT is almost ideal to demonstrate the benefits of modular control, as the DT-VGT is a very clearly defined mechanical module that lends itself to modular control.

10.4 Generic I/O

One of our main sub projects have been the development of a flexible I/O system based on FPGA's (Field Programmable Gate Arrays). It is this technology that have enabled the combination of flexibility and compactness necessary to implement the small, yet powerful and flexible controller platforms used for the GEECONs.

The I/O system allow us to transfer a lot of the functionality usually implemented with dedicated hardware, to a reconfigurable digital system, that can be configured and reconfigured to handle an extreme variety of I/O functions, without any physical changes.

Our I/O system can not completely remove the need for dedicated hardware, but reduces the amount and complexity of the dedicated hardware to a very modest level compared to conventional I/O systems.

The flexibility of our I/O system enable us to interface to virtually any electrical system, reduce development time dramatically, allow us to optimize the I/O logic to the specific task, and allow us to alter, debug and improve the interface without changing the physical configuration.

The I/O logic in our system is designed as a modular system based on the hardware description language VHDL. This approach enable us to move I/O components across peripheral bus systems and computer platforms, so I/O components can easily be reused in very different applications once they have been developed. This principle have been demonstrated in parallel projects, where our I/O system have been used with the *Industry Pack* and PCI peripheral busses.

We find our approach to I/O very interesting for research and development projects and for any environment that needs rapid prototyping of I/O systems.

10.5 Network technology

As discussed in chapter 3, the bandwidth and latency of the *back bone* network chosen for our distributed control system is crucial to its success, and future applications.

In order to benefit from state of the art network technology, we have investigated IEEE-1394 or FireWire as a possible technology for the *back bone* network. We found the technology well suited for this application, as it supports isochronous transfer rates and latencies of $800\text{Mbps}/250\mu\text{s}$, which is substantially better than our $2\text{Mbps} / 500\mu\text{s}$ minimum requirement. Unfortunately it turned out that not all hardware and software implementations utilize the potential of the technology in an efficient way. The Firewire interfaces available for our CPU platform turned out to be premature, and were abandoned in favor of the well proved ARC-net.

Due to our previous experience with ARC-net, its integration into this project have not caused any surprises. With bandwidths of up to 10Mbps and even and deterministic access for the nodes to transmit, it is quite well suited for our application. Its latency depend on various parameters

related to the number of nodes and the protocol implemented by the controller software, so care must be taken to implement protocols that keep the latency sufficiently low.

Though ARC-net is substantially slower than FireWire, it still has excessive bandwidth in comparison to our estimates of the needs for practical industrial applications. The additional bandwidth can be used to ensure compliance with demands in excess of our estimates, or it can be set aside for future integration of e.g. external sensors. In order to ensure the integrity of the network, such sensors must be integrated with a GEECON and the communication protocols must be designed to accommodate isochronous data which is not directly related to the motion control.

10.6 Reference model

The controller reference model developed in order to provide a framework for description and discussion of our work, have worked quite well. As it is not intended as a design or implementation model, it has not resulted in any physical experience, but is has proved to be very useful when discussing controller technology with our partners and associates.

As our reference model is inspired by previous experience with controllers, we are not surprised by the fact that our controller architecture has a strong resemblance with the reference model, and it is quite possible that our reference model could be expanded into useful design- or even implementation models.

10.7 Hardware / Software co-design

We have been extremely impressed with the development method and the design possibilities offered by our use of reconfigurable logic in the I/O system. We have recognized that this development method — co-design of hardware and software — may have a profound impact on the way computer applications are developed. Reconfigurable logic allow the application designer to increase the system performance by implementing various functionality in hardware. Beside I/O applications these could be:

- Specialized memory, e.g. multi port, queues, and monitors.
- Specialized logic functions, e.g. endian conversion.
- Algorithms, e.g. transformations and filtering.

Complex reconfigurable logic devices like FPGA's have been available for roughly a decade, and they have become very common in modern computer components. We are surprised that hardware/software co-design does not seem to be in common practise with application developers, but that the use of reconfigurable logic seems to be reserved for electronic engineers. We speculate that the traditional hardware/software partitioning of computer systems has become too deeply rooted with the research, and education communities. The partitioning may indeed have

become a preconception that inhibits a possible convergence between hardware and software development.

Chapter 11

Conclusion

During this project we have reached our main goal by developing a generic embedded control node (GEECON), that is able to transform a wide range of industrial mechanisms into *intelligent* modules that can be incorporated in a modular controller architecture.

We have demonstrated the technology by integrating it with two very different robot modules. The primary demonstration module is the experimental, hydraulic powered, 3 degree of freedom (DOF), parallel kinematic module (PKM), known a double tripod variable geometry truss (DT-VGT).¹ The secondary module is the Mitsubishi PA-10 conventional 7 DOF robotic arm.²

The ability to integrate robotic modules controlled by GEECONs have been demonstrated by connecting the PA-10 to the DT-VGT, and using the aggregated 10 DOF robot to spray paint a wheel barrow.³

We have reached our goal using a reactive project model, that have allowed us to adapt to a very dynamic and challenging project environment.⁴ The reactive model have enabled us to benefit from corporation with an array of partners, ranging from engineering students to the participants of the EU project DockWelder.

This project is partly financed by the EU project *DockWelder*, where the GEECON for the DT-VGT represent an important contribution. Our commitment to deliver working prototypes for DockWelder, have emphasised the engineering aspects of the project, and helped us to focus on usability throughout our work.

¹See chapter 7

²See chapter 8

³See chapter 9

⁴See “Foreword”

Key technology

We have obtained our results from integrating 3 key technologies: a CPU, a network and an I/O system to form the platform for a GEECON. The architecture, technologies and components have been chosen or designed to comply with demands, parameters and suggestions, derived from our overall experience and assumptions about practical industrial mechanics.⁵

Our main contribution to the the GEECON technology is the flexible I/O system, that allow a significantly better tradeoff between performance, compactness and flexibility than conventional embedded technology.⁶

The flexibility have been demonstrated by engaging the GEECONs and the I/O technology in a variety of projects and experiments in- as well as outside this project. The compactness have been demonstrated by implementing GEECON platforms for the PA-10 and DT-VGT, which only measures $7 \times 14 \times 5$ cm.⁷ The performance is evident from the specifications of the sub-components, but have also been partly demonstrated in an external project, where a GEECON prototype was used as controller, I/O interface and preprocessor for an experimental ultrasonic scanner, demonstrating I/O bandwidths up to 60MHz.

Hardware/software convergence

The reconfigurable logic used in the I/O system is closely integrated with the CPU, its functionality is specified in a language quite similar to conventional programming languages, and it's configuration can be changed or updated at any time. This effectively blurs the traditional sharp border between software and hardware, as it becomes possible for the application developer to change the system hardware as easily as he changes the software. It is even possible to let the software specify or change the hardware dynamically, or specify program-executing state machines, e.g. CPU's for the reconfigurable hardware.

While our GEECONs still rely on an application specific hardware layer for e.g. signal conditioning and conversion, the way this layer is interfaced to software has changed dramatically. We have used the reconfigurable hardware to adapt the functionality offered by the static hardware to the architecture of the CPU, as well as to our preferred low level software architecture.

The co-development of software and I/O logic have allowed us to interface the control software of the GEECONs to the hardware of e.g. the DT-VGT in an optimal way. Both the software and the application specific hardware (signal conditioning etc.) used for this project is significantly simpler, more compact and elegant than their counterparts in previous projects.

The hardware/software partitioning of traditional computer systems arise because software development has traditionally been much more dynamic than hardware development. The inclusion

⁵See chapter 3

⁶See chapter 5

⁷Without enclosure etc.

of reconfigurable logic changes that balance. Application designers who are adept at interdisciplinary development in the fields: software, digital design, and electronics; can use reconfigurable logic to remove the border between hard- and software, seriously changing the way we look at computer systems and applications.

Systems integration

Our I/O architecture enable the GEECONs to be interfaced to virtually any mechatronic system by implementing an application specific *daughter board*.⁸ The daughter board provides the necessary signal conditioning and conversion between the reconfigurable logic of our I/O system and the mechatronic system.

The use of reconfigurable I/O logic gives a great degree of freedom to the daughter board design, which simplifies the design process, and enables rapid development of daughter boards as well as experimental interfaces for different applications. During this project, daughter boards and experimental interfaces have been developed for the DT-VGT, the PA-10, the ultrasonic scanner mentioned above, as well as a number of other minor projects and experiments.

The software developed to demonstrate the DT-VGT and the PA-10 modules feature position control of the individual actuators, and allow the actuators to be controlled on-line through the GEECONs network, or off-line by downloading joint path descriptions to the GEECONs in advance.

The ability to integrate *intelligent* mechanical modules into aggregated robots were demonstrated by connecting the PA-10 and the DT-VGT, and letting their GEECONs control them in synchronised motion that allowed the aggregated robot to spray paint a wheel barrow.⁹

Contribution

This project have contributed to the knowledge, understanding and research in robotics and automation in a number of ways.

The development of a practical controller for the DT-VGT mechanism is an important contribution to the further development of variable geometry trusses, where the robotics community of Odense holds a leading position.

We have introduced a generic embedded control node (GEECON) that is sufficiently compact, powerful and flexible to be embedded into, integrated with and control most practical industrial mechanisms. As we have demonstrated, the GEECON can be used to transform existing industrial mechanics into *intelligent* mechanical modules that can easily be integrated into complex modular systems.

⁸See section 4.3

⁹See section 9.2

The GEECON technology has been introduced in various contexts related to robotics as well as other aspects of embedded control, and it is being adopted by the robotics group at university of southern Denmark as the preferable low level controller technology for experimental robotics.

The ability to use existing mechanics as building blocks for modular robots increase the feasibility of experimental and temporary robotic systems. This will benefit research and development (R&D) of industrial robotics, as well as industrial use of robots in temporary or dynamic production lines.

The demonstration of hardware/software co-design in the GEECON I/O system is an illustration of current and emerging possibilities in computer engineering. It underlines the great importance of interdisciplinary insight to efficient computer systems development. At the same time it points to reconfigurable logic and HW/SW convergence as a very important area for R&D as well as education within the field of computer systems engineering.

This project span a wide area of topics, technologies and disciplines, which have successfully been brought together to form a working modular control system for industrial mechanics. The control system is an enabling technology for further local R&D of variable geometry trusses, and is a major contribution to the ambitious EU project DockWelder. The project is in itself a demonstration of the benefits of interdisciplinary R&D in computer systems engineering, and is an important contribution to the local computer systems engineering environment.

Chapter 12

Future work

In this chapter we will discuss the opportunities for future research and development, while we refer to the *Future work* sections of the technical chapters (4 to 9) for descriptions of the minor technical issues.

12.1 Generic embedded control nodes (GEECONs)

We are quite satisfied with the current hardware architecture of the GEECONs, and estimate that it can support relevant projects for some years to come. As Orsys GmbH continues to support and upgrade the Microline range of CPU modules used in the current GEECON, the supply of CPU modules does not appear to become a problem. It might however be desirable to become independent of Orsys as the only supplier of CPU modules, which can be accomplished by developing an independent Microline compatible CPU module, or by designing a new generation of GEECONs with e.g. integral CPU.

The current trend in reconfigurable logic is to integrate powerful CPU's and large RAM blocks with increasingly complex gate arrays, on the same IC. This development makes it possible to implement most of a GEECON in a single IC, with power supply and signal conditioning being the only external components. In light of this trend, we recommend considering this technology for the next generation of GEECONs, as it can make the GEECONs substantially smaller without compromising the flexibility and performance of the current architecture.

As described in chapter 9, the GEECONs will be integrated with the mechanical modules used in the DockWelder EU project during the spring of 2003. The robotics group of university of southern Denmark is currently planning to participate in a number of industrial projects, where the GEECONs will be integrated with various placer mechanisms e.g. mechanical lifts. In addition to external projects, we have plans to equip previous robotic prototypes and experimental systems with a GEECONs in order to ensure compatibility between the experimental systems of

the local robotics research. These upgrades will be out sourced to student, in order to provide education in low level control and GEECON know how at the same time.

12.2 Network

As there are obvious advantages to using a faster and more popular technology than ARC-NET, we recommend that the network technology is reconsidered when a new generation GEECONs are designed. We recommend taking a second look at FireWire, but as there is a tremendous development in computer networks, other options may prove interesting at that time.

12.3 Central controller

The central controller currently holds the greatest potential for improvements, as it currently exists only as discrete software components. An ad-hoc central controller that supports on-line control of a modular robot, will be operational within the summer of 2003. In order to harvest the full potential of our modular architecture, we need a central controller that can interact with the GEECONs with respect to configuration as well as motion and process.

We recommend that the local research and development of controller technology is intensified, and that local researchers engage in cooperation with organisations that posses relevant high level controller technology and experience.

12.4 Reconfigurable hardware

The development of a flexible I/O system based on reconfigurable digital hardware, has been a very important sub project for this work, and it has proven interesting to other fields than modular control.

In this project we have relied on static analog hardware for much of the necessary signal conditioning. As reconfigurable analog hardware is currently emerging as a feasible technology, it will be interesting to research the potential of this technology as well.

We find the possibilities offered by reconfigurable hardware, to allow hardware- and software-design to converge extremely interesting. This technology have proven particularly useful in I/O systems, but might also affect other aspects of computer systems development. Many trends point toward the future importance of reconfigurable hardware, and we strongly recommend further research and development in this field, both in relation to control technology, and as an area in itself.

Part I

Appendix

Appendix A

Danish summary

Baggrunden for dette projekt er det lokale udviklings og forskningsarbejde der siden starten af 1990'erne har fundet sted inden for områderne:

- Generel bevægelsesplanlægning til industrielle robotter.
- Specialbyggede robotter til anvendelse i bla. skibsbygning.

Det har vist sig at den mest effektive måde at designe og konstruere avancerede robotter med mange frihedsgrader, er at bygge dem op vha. forskellige mekaniske *moduler* der hver især har en bestemt funktion. Nogle af disse moduler kan være kommersielt tilgængelige enheder som f.eks. en almindelig industri robot, mens andre moduler kan være specialbyggede til et bestemt formål, som f.eks. Meganic ApS' DT-VGT. Komplekse sammensatte robotter kan så nemt fremstilles til forskellige formål ved at sætte de grundlæggende mekaniske moduler sammen på forskellig vis.

Formålet med dette projekt er at udvikle en generel indlejret styredatamat, der kan indbygges i de forskellige mekaniske moduler, og dermed transformere dem til intelligente selvstændige robotmoduler, der kan koordineres af en central datamat blot ved at samle dem og forbinde dem til et fælles kommunikations netværk.

Rapporten begynder med at beskrive og sammenligne relevante områder af modulær robotteknologi, generelle robotcontrollere og indlejrede robotcontrollere (kapitel 2).

I kapitel 3 foreslås en passende arkitektur til en generel modulær robotcontroller, der baserer sig på et netværk af små fleksible og kraftfulde indlejrede datamater, der tager sig af lokal styring af de enkelte mekaniske moduler. Al overordnet styring af den sammensatte robot håndteres af en central enhed, og de forskellige enheder kommunikerer via et fælles netværk. Den centrale enhed ligger uden for vores emneområde, da denne funktion kan varetages af tidlige udviklede robotcontrollere som f.eks. Odense Stålskibsværfts OMC controller. Der opstilles dernæst krav og anbefalinger til forskellige aspekter af de indlejrede datamaters specifikationer, fleksibilitet og ydeevne. Disse krav udledes fra vores kendskab og forventninger til industrielle robotsystemer.

I kapitel 3 beskrives den overordnede udvikling af de indlejrede datamater der skal fungere som platform for vores system. Datamaterne bygges af tre hovedkomponenter:

- Et kommersIELT tilgængeligt CPU modul, der senere kan opgraderes mht. ydeevne mm.
- Et *Motherboard* der rummer strømforsyning, netværks interface og Input/output logik.
- Et *Daughterboard* der rummer de specifikke kredsløb der er nødvendige for at sammenkoble datamatet med en bestemt mekanisme.

Det I/O system vi har udviklet til den indlejrede datamat er nøglen til at opnå et meget fordelagtigt forhold mellem størrelse, ydeevne og fleksibilitet. Vores arbejde med I/O systemet er derfor beskrevet udførligt i kapitel 5.

Den konventionelle måde at realisere et fleksibelt I/O system på er ved at basere den på I/O hardware moduler, som vi kender dem fra ISA, PCI, VME, Industry Pack etc. Denne praksis giver ganske vist stor fleksibilitet, men den kræver meget plads og er derfor hæmmende for udviklingen af en generel indlejret datamat.

Vi erstatter de konventionelle I/O moduler med programmerbar logik, i form af en *Field Programmable Gate Array* (FPGA), der placeres på datamatens *motherboard*. FPGA'en kan konfigureres til at udføre stort set alle digitale funktioner, og kan dermed konfigureres til at fungere som I/O interface mellem CPU modulet og den ydre verden.

FPGA'ens grænseflade er digitale signaler, så et vist omfang af signaltilpasning og konvertering er nødvendig for at forbinde FPGA'en til den ydre verden. Ved at placere de nødvendige kredsløb til dette på datamatens *daughterboard*, kan vi nøjes med at udvikle et forholdsvis simpelt daughterboard til hvert mekanisk modul vi ønsker at tilkoble vores indlejrede datamat.

FPGA'en konfigureres i et hardware specifikations sprog (VHDL) der minder om kendte programmeringssprog, og også tillader at udvikle modulære designs. Ved at udnytte dette opbygges et *bibliotek* af digitale I/O funktioner der kan genbruges i forskellige sammenhænge.

I kapitel 6 gennemgås vores arbejde med netværks teknologi. Først gives en oversigt over populære og relevante netværksteknologier, og dernæst gives en mere grundig gennemgang af de to teknologier vi har arbejdet med i dette projekt.

Da netværket skal overføre data der er tidskritiske, og samtidigt have så stor kapacitet som muligt har vi først forsøgt at anvende IEEE-1394 eller FireWire, da det understøtter tidskritisk datatransmission med op til 800Mbps. Firewire er umiddelbart meget velegnet til dette projekt, men vi var nødt til at fravælge det fordi det firewire modul af det programmel der var til rådighed for vores CPU moduler ikke var tilstrækkeligt effektivt.

Som erstatning til FireWire har vi valgt ARC-net, der er et pålideligt og gennemprøvet token bus netværk til industrielt brug. Vi har en del erfaring med ARC-net, og kunne nemt implementere en passende løsning.

I kapitel 7 beskriver vi hvordan vi har koblet Meganic ApS DT-VGT modul sammen med vores indlejrede datamat. Dette arbejde indbefatter bla. udvikling af en passende forstærker til modulets hydraulikventiler, overvejelser om strømforsyning, implementation af passende *daughter-board*, konfiguration af FPGA, udvikling af demonstrations programmel, genbrug af regulering fra et tidligere projekt, og overvejeler om modellering af modulets hydrauliske system.

I kapitel 8 beskrives det hvordan en Mitsubishi PA-10 robotarm kobles til en indlejret datamat vha. et ekstra ARC-net modul og udvikling af passende demonstrations programmel.

I kapitel 9 beskrives integrationen af en DT-VGT og en PA-10 robot, der begge styres af en indlejret datamat. For at demonstrere systemet har forskellige lokale forskningsprojekter bidraget til at en opstilling hvor den sammensatte robot kan bruges i en industriel proces, i dette tilfælde at sprøjtemale en trillebør.

Vha. en CAD model af trillebøren og malersprøjen har robotgruppen ved Mærsk Mc-Kinney Møller Instituttet for produktionsteknologi (MIP) først beregnet en passende bane for malersprøjen. Denne bane er sammen med en CAD model af robotten brugt til at beregne baner for robottens individuelle aktuatorer. Disse baner er overført til de respektive indlejrede datamater i den sammensatte robot, og datamaterne får derefter deres værts mekanismer til at udføre den samlede bevægelse synkroniseret af en central computer.

Kapitlet afsluttes med en beskrivelse af en tilsvarende men mere avanceret demonstrationsopstilling der i øjeblikket forberedes til EU projektet DockWelder.

Kapitel 10 er en evaluering og diskussion af de væsentligste erfaringer fra projektet der er overvejende positive, med meget få betænkeligheder angående det udviklede system.

I kapitel 11 konkluderer vi at projektet er vellykket, og hæfter os især ved vores brug af konfigurerbar logik som den nøgle teknologi der har gjort det muligt at fremstille en tilstrækkelig kompakt, fleksibel og kraftig indlejret datamat. Samtidigt erkender vi at rekonfigurerbar logik har et stort potentiale for at skabe konvergens mellem programmel og elektronik, hvilket kan give store fordele for udviklingen af applikationer inden for mange aspekter af dataknologi.

Appendix B

Published papers

During our work with modular controllers, we have contributed to a number of published papers, as described in the following pages. The articles themselves are placed in Adobe PDF format on the accompanying CD-ROM.

B.1 Design of Double-Octrahedral VGT Manipulators

Published: VDI BERICHTE 1427 “Neue Maschinekonzepte Mit Parallelen Strukturen Für Handhabung Und Produktion”, 1998

Authors:

- Dipl. Ing. O. G. Jakobsen
- Dipl. Ing. S. A. Larsen
- M. Sc. A. S. Sørensen
- M. Sc. N. J. Jacobsen

Abstract: This paper presents a 19-degrees of freedom prototype robotic manipulator built with the purpose of demonstrating the potential of the parallel mechanism called a Variable Geometry Truss (VGT). The VGT mechanism is characterized by a high rigidity-to-weight ratio and has its advantages in long-reach or high-payload tasks. All previous VGT designs are found within the field of space or nuclear waste operations. However, this manipulator is aimed for conventional industrial purposes. This paper presents the intended application of the manipulator, the VGT technology, workspace studies, a unique zero-offset joint, rigidity analyses, and the control strategies.

My contribution: This paper features a description of my very first attempt at a modular embedded control architecture, based on conventional industrial computers.

File: Papers/vgtvdi.pdf

Co. author statement:

As co-authors, we acknowledge that Anders Stengaard Sørensen have contributed to the paper as stated above.

Ole Grå Jakobsen

Niels Jul Jacobsen

B.2 A development of parallel robotic modules for long reach applications

Published: Proceedings of the 32nd ISR (International Symposium on Robotics), 19–21 April 2001

Authors:

- M. Sc. A. S. Sørensen
- As. Prof. Henrik G. Petersen
- Dipl. Ing. O. G. Jakobsen
- M. Sc. N. J. Jacobsen

Abstract: In this paper, we present our development towards variable geometry truss (VGT) submodules containing embedded processors for real time control. We first discuss how prototype modules were developed and the experiences we have got from performance studies of that. Next, we present the design phase of new modules where we use kinematic and dynamic computer models of the modules together with Finite Element packages to maximize the kinematic working area of the module taking the load on the joints and links into account. We then discuss the design and development of the small and flexible high performance embedded processors. We discuss in some detail the motivation for having such processors and the choice of processor by the European Commision aimed at enabling automation of unhealthy welding tasks in confined steel structures in the dock area in ship building.

My contribution: In this paper I describe the advantages of integrating embedded computers with the DT-VGT's as well as other mechanical modules, and go on to describe the technology we plan to use to accomplish a sufficiently compact and flexible design.

File: Papers/parallel.pdf

Co. author statement:

As co-authors, we acknowledge that Anders Stengaard Sørensen have contributed to the paper as stated above.

Henrik Gordon Petersen

Ole Grå Jakobsen

Niels Jul Jacobsen

B.3 Towards the industrial usage of parallel kinematic modules in a fully modular robotic manipulator

Published: Fundamental Issues and Future Research Directions for Parallel Mechanisms and Manipulators October 3-4, 2002, Quebec City, Quebec, Canada

Authors:

- M. Sc. A. S. Sørensen
- As. Prof. Henrik G. Petersen
- Dipl. Ing. O. G. Jakobsen
- Dipl. Ing. J. Steinicke

Abstract: One of the main advantages of parallel robots is that they can be applied as modules in long-reach (snake-like) robotic manipulators enabling robotic automation of unhealthy work in confined spaces. However, before parallel kinematic robotic modules (PKRMs) can be widely used in such applications a variety of technological tasks must be accomplished such as design issues, precise model based calibration, modular control and process control. It is the purpose of this paper to address our work within each of these tasks aimed at developing robust reconfigurable PKRMs for long reach welding applications within e.g. shipbuilding.

My contribution: In this paper I describe the demands the DT-VGT modules puts on an embedded controller, and describe how an I/O system based on configurable logic can accommodate many of these.

File: Papers/des_opt_1.pdf

Co. author statement:

As co-authors, we acknowledge that Anders Stengaard Sørensen have contributed to the paper as stated above.

Henrik Gordon Petersen

Ole Grå Jakobsen

Jakob Steinicke

B.4 A development of modular robots for flexible robotic manufacturing units

Published: Proceedings of the 33nd ISR (International Symposium on Robotics), 2002

Authors:

- M. Sc. A. S. Sørensen
- As. Prof. Henrik G. Petersen

Abstract: With todays rapidly changing consumer driven market, the need for robotic production units that are capable of being modified fast and efficient has increased tremendously. An obvious way to fulfill this need is to use truly modular units as building blocks for the robotic production units. Using this approach, one can then think of these production units as consisting of robotic manipulators that are designed in advance in the office based on the available modules and the tasks the manipulators are going to perform. The modular subunits can be sub-sets of a long robotic arm, but they may also be moving platforms, such as fork-lifts or other vehicles, or atomic one degree of freedom units. It is the purpose of this paper to present our development of core technologies for industrially robust robotic modules. More specifically, these technologies are embedded controllers for the modules and a central kinematic controller that mirrors the modular idea of the underlying modular mechanical and electronic hardware of the robotic manipulator.

My contribution: In this paper I describe the benefits of modular control and introduce the layered reference model. I move on to describe the components of a modular controller in terms of the layered reference model, as well as the challenges posed by implementing such a system and the means to overcome them. Finally I give a description of our prototype controller and how it is integrated with a DT-VGT.

File: Papers/isr2002.pdf

Co. author statement:

As co-author, I acknowledge that Anders Stengaard Sørensen have contributed to the paper as stated above.

Henrik Gordon Petersen

Appendix C

Goal analysis

This chapter indicates the original analysis of the project goals. The analysis was performed during the fall 2000 and is included here to motivate the project breakdown of appendix D.

C.1 Overall task description

The vision for this area of work is to develop a system of relatively few robot modules, based on parallel as well as serial mechanisms, enabling a large number of different manipulators to be constructed by putting the modules together in different ways. Modular control technology will enable easy connection, configuration and calibration of complex manipulators.

As part of this vision, this Ph.D. project is supposed to suggest technologies and principles for distributed modular control computers. Experience from previous tests show the need for physically small computers, closely coupled to the actuators in the various modules. The basic idea of this project is to investigate and test methods and technologies for creating a control unit capable of:

- Being integrated with one or more electrical, pneumatic or hydraulic actuators and control them in a closed loop, essentially making the actuators *smart*.
- Participate as a node in a network connecting a number of *smart* actuators into a network, enabling them to exchange relevant static and dynamic information.
- Presenting the system as a whole, with as little coordination from a central node as possible.

The origin of the project is the past development on VGT¹ robot technology, performed at AM-ROSE A/S. The project is meant to result in a demonstration of a VGT robot performing controlled movements.

¹Variable Geometry Truss

C.2 Conceptual basis

During the last 10 years, Odense University have participated — directly and indirectly — in the development of motion planners for complex robotic manipulators, consisting of an aggregation of a standard industrial robot and various custom build gantry positioning systems. Giving a total of up to 12 degrees of freedom.

A common factor in all thesee projects have been the enormeous amount of work needed by mechanics, electricians, engineers and programmers, to integrate the various actuators, sensors, interfaces and other elements into a working, documented system that could be controlled by the computer running the motion planner software.

Modularity

Object oriented metods has led to many improvements on the higher levels of software, treating each basic mechanical element as an instantiation of a common class. Low level software (device drivers), hardware interfaces, wires, motor controllers, sensors etc. are however largely designed and implemented the old fashioned way: Treating the whole installation as a sigle complex integrated system.

The benefits of extending the object oriented — or modular — view to the mechanical elements themselves are particularly clear when considering VGT modules. One of the main benefits of VGT's are that a number of VGT modules can be assembled into manipulators, which workspace, speed and handling capabilities can be varied almost indefinitely, depending on the number, type, and configuration of the basic modules.

Modular control

By embedding a computer in each VGT module, each module can take care of it's own low-level control, leaving the central controller to specify the physical motion to be performed, via a standardized network interface. In this way, each mechanical module can be integrated with the system, simply by connecting it to a common network and a power supply. The principle can be extended to all other mechanical elements of a (robot) installation. Compatible computers can be embedded in and interfaced to off the shelf industrial robots e.g. the Mitsubishi PA-10, or customized elements e.g. a gantry positioner.

Configuration

Having a computer with permanent storage in each module, makes configuration af the central motion planner much easier, as the motion planner can interrogate the network to find out which

modules are present, what their order is, and what their kinematic and dynamic parameters are. Based on these informations, the motion planner can configure its own mathematical model of the robot without intervention of an operator and without the risk of mismatches between model and reality.

Experience

To gain experience, we have already tested embedded distributed controllers on VGT prototypes. The main lessons were the importance of fast real-time networks, and the necessity to use computers that are substantially faster, smaller and cheaper than what is currently available for industrial control.

Future technology

By using computing and network technology developed for professional multi media purposes i.e. DSP's and Firewire, and by developing a highly reconfigurable I/O subsystem based on programmable components rather than physical modules, it will be possible to meet all the demands derived from present experience: Size, speed and cost, while maintaining the flexibility necessary to integrate the system with a wide variety of actuators and sensors.

C.3 Goals

The project has a number of different goals, which are compatible under the right conditions, but can be conflicting under time pressure.

Product goals

Prototype platform: Developing a prototype platform for a control module is essential to the goal of the overall project description. The platform be the basis of other product goals, but can also be used in other projects and activities.

Prototype on DT-VGT: The next logical step to meet the overall goal is to integrate control modules with DT-VGT mechanisms, and demonstrate that the module can control the DT-VGT, effectively making the DT-VGT low-level autonomous.

Interface for motion planner: To make the autonomous DT-VGT useful, it must be able to accept commands from a motionplanner. During this project this will be restricted to an off-line motion planner.

Prototype on other mechanism: To demonstrate and test the ability to control heterogeneous mechanisms, it is necessary to integrate the control module to another kind of mechanism, that is able to work together with the VGT modules.

Demonstration of integrated heterogeneous system: The final goal is to demonstrate a heterogeneous manipulator working as an integrated system.

Functional goals

Supplement present controller technology: While the attention of the product goals are limited to a few mechanisms, the technology identified and developed in the project, should be able to supplement or even replace existing controller technology.

Boosting VGT technology: As most standard mechanisms are already integrated with a suitable *native controller*, the most likely area of commercial applications are as VGT controllers. As a consequence, it is important that the technology is developed with this specific mechanism in mind, as the availability of a convincing control system is adamant for marketing the VGT technology.

Boosting activities at MIP: Many of the activities at MIP are related to control of robots and the means to do so. These activities should benefit from the technology of the project.

Education: The participants of the project must gain additional knowledge and experience in the field of control systems and computer systems engineering.

Integration with Dockwelder: The concepts of this project will be continued in Dockwelder. To get the most out of both projects, this project should adapt to the structure of Dockwelder.

Administrative goal

Completing Ph.D. thesis: One of the dominant goals is to finish my Ph.D. thesis. This will allow me to pursue an academic career and to get interesting — possibly better paid — jobs in industry.

Keeping the Feb. 2000 deadline: SDU stops paying me to work on the thesis by Feb 2002.

Publishing papers: A Ph.D. thesis is supposed to yield a number of scientific papers due to the *Publish or Perish* principle. Partly to sustain an image of scientific productivity, partly to enhance the possibility of an academic career.

Other goals

One of the reasons for the Maersk Institute to invest in this project, is to enable me to continue working at the institute after the thesis. This is not exactly a goal for this project, but it is nevertheless intertwined with the project in many ways.

C.4 Criterions for success

As there are a number of concurrent goals for the project, it is essential to sort out what the actual criterions for success are. Unfortunately, this depends largely upon viewpoint. Four different viewpoints have been identified, as shown in figure C.1.

Goal	Viewpoint			
	Academic	Industrial	Maersk institute	Personal
Completing Ph.D. Thesis	3	0	3	2
Keeping Feb 2002 deadline	2	0	1	1
Publishing papers	3	0	2	1
Generic control platform	1	2	2	2
Integration with DT-VGT	1	3	3	3
Integration with PA-10	1	2	2	1
Interface for motion-planner	1	3	3	3
Demonstration of heterogeneous system	1	2	2	3
Supplement present controller technology	0	1	2	2
Boosting VGT technology	0	3	1	3
Boosting activities at the Maersk Institute	0	1	3	2
Education in related fields	1	1	2	2
Integration with dock-welder	0	3	3	3

0 = Unimportant 1 = Interesting 2 = Important 3 = Essential

Table C.1: Goals prioritized by viewpoints

Academic viewpoint

In a pure academic environment, the criterion for success would be:

Minimal: To complete an acceptable thesis, while publishing a few papers.

Supplementary: Having little or no delay.

Benefits: Demonstration practical applications of the project, and educating a few people.

Industrial viewpoint

Taking the viewpoint of our industrial partners, success might be defined as:

Minimal: Demonstrating the ability to control DT-VGT's, by way of a motion-planner, thereby boosting the VGT technology.

Supplementary: Demonstrating the ability to control heterogeneous mechanisms, thereby paving the way for dockwelder.

Benefits: Creating supplementary technology for existing controllers, educating students and increasing MIP activities in related areas.

Maersk Institute viewpoint

As the Maersk Institute is a complex matter, it is not easy to specify this viewpoint. Assuming that the institute acts as a bridge between industrial and academic interests, and allowing for some special interests, success might look like this:

Minimal:

- Completing the thesis.
- Enabling dockwelder by interfacing VGT-modules to a motion-planner.
- Increasing related activities at MIP

Supplementary:

- Publishing a number of papers.
- Educating students in related areas.
- Demonstrating the generic aspects of the platform by controlling heterogeneous mechanisms.
- Integrating the existing PA-10 robot to the system.

Benefits:

- Keeping the deadline.
- Augmenting MIP research in parallel kinematics and dynamics by boosting VGT technology.

Personal viewpoint

Minimal:

- Demonstrating the value of 3-5 years of work in this field, by getting a heterogeneous mechanism to work.
- Enabling industrial applications for DT-VGT's.

Supplementary:

- Getting the benefits of a Ph.D. degree.
- Increasing 'computer systems engineering' activities.
- Supplementing existing controller technology, creating more applications for the modular controller.

Benefits:

- Avoiding the problems of breaking the deadline.
- Getting the benefits of having published a range of papers.

Appendix D

Work Breakdown Structure Analysis

This chapter indicates how the project was analyzed and subdivided into various smaller parts. The analysis was performed during the fall of 2000. As we were never able to expand our project group, the need for documentation of this process have been limited, leading to a low level of detail in this chapter.

D.1 Functional structure

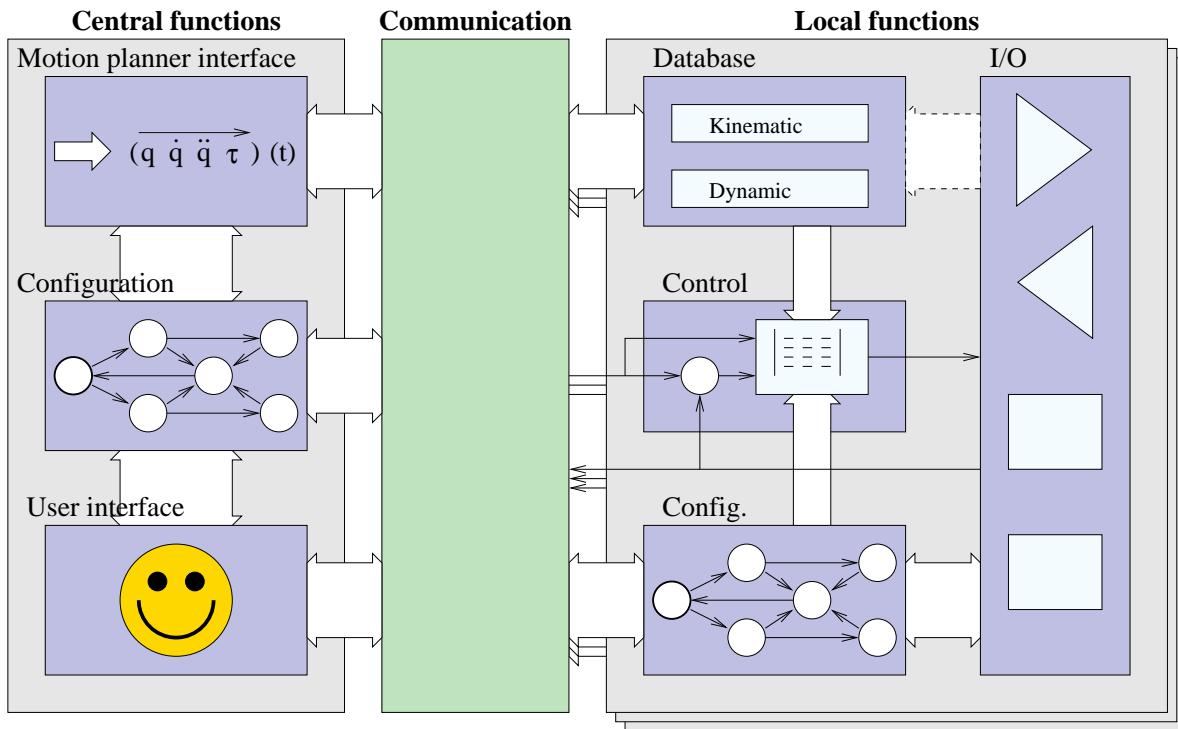


Figure D.1: Functional breakdown of project

Modular control system

The main function of the system is to act as a modular control system for generic industrial mechanics.

Communication

The communications system connecting control modules and central controller

Local configuration

The system controlling the mode of each module eg. On, Off, On-line, Error etc.

Local control

The control systems in charge of actuators an individual modules.

Local I/O

The I/O system connecting the local computer to the host systems.

Local database

Holding relevant kinematic and dynamic information about the individual robot modules.

Central configuration

Interface for motion planner

Central user interface

D.2 Systems structure

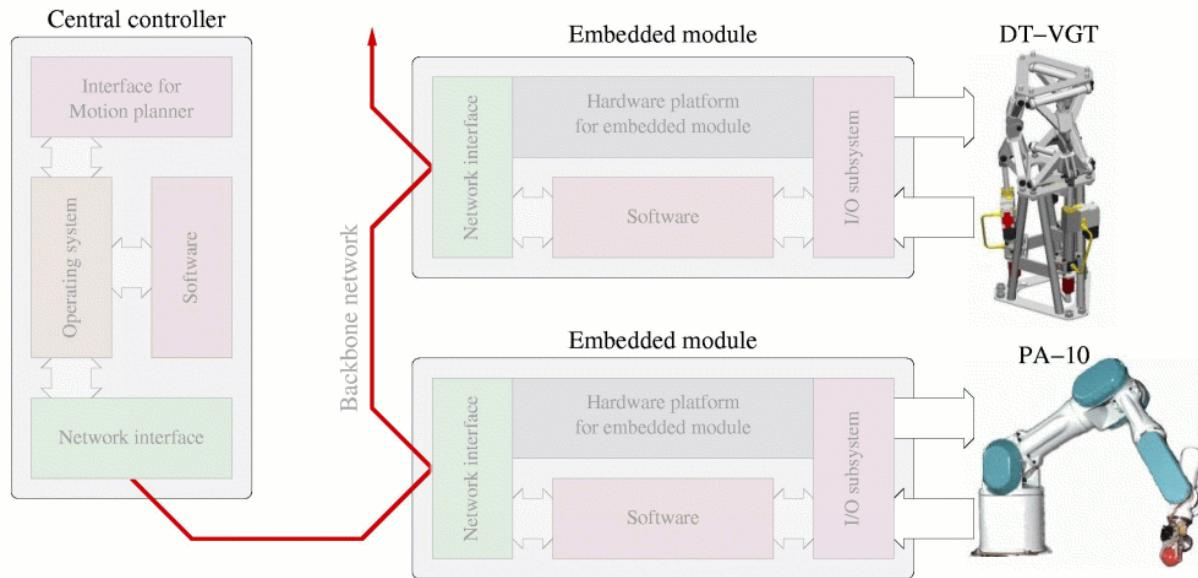


Figure D.2: System oriented breakdown of project

Overall system

This is the product oriented counterpart of the main function as *Modular control System*, described in section D.1. The two concepts represents different views of the same thing, and can be used interchangeably.

Embedded module

The embedded modules are the central concept of the whole project, and the most important product goals. As indicated in figure D.2, the embedded module can be subdivided into several logical parts.

Central controller

The central controller is thought of as a conventional computer, running a conventional operating system. It is equipped with an interface, connecting it with the embedded modules via. the backbone network.

DT-VGT

Interfacing the module to the DT-VGT mechanism is a main feature of the project. It involves:

- An I/O subsystem, interfacing the sensors and actuators of the DT-VGT to the host computer.
- Control algorithms for the DT-VGT.
- Physical integration.

PA-10

Interfacing to the PA-10 involves the same overall aspects as interfacing to a DT-VGT, but the particulars are different.

D.3 Project Phases

Figure D.3 shows the anticipated flow of the project, divided into 4 phases, with the possibility of iterations on many levels.

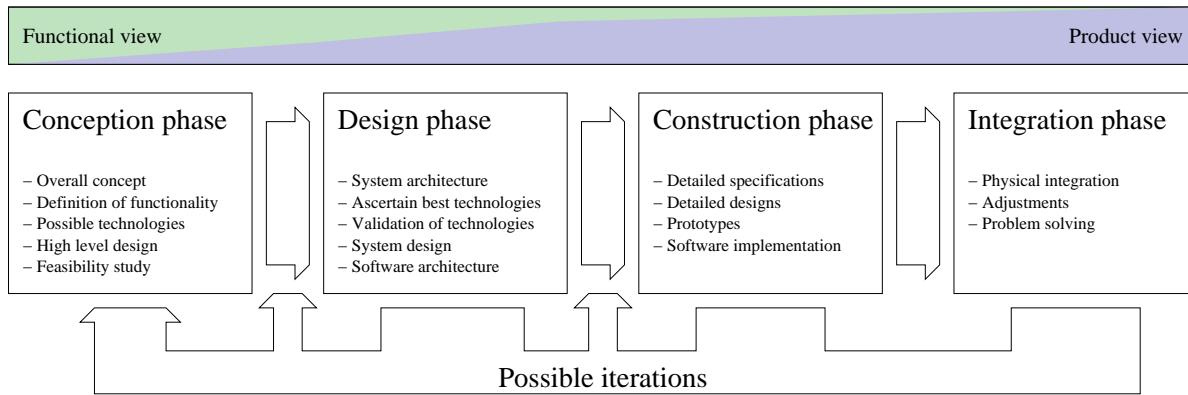


Figure D.3: Project phases and flow

D.4 Areas of effort

To enable consistent management of the project, the areas of effort are identified. Naturally, the main effort is in the technical aspects of the project, but it is vital to the success of the project that other areas are not neglected.

Technical

The natural areas of effort within the technical domain, coincides with both the functional and system oriented breakdown of the project. As shown in figure D.3, the project becomes more systems oriented toward the end, while the systems are not clearly defined towards the beginning.

To get a clear picture of the relationship between functions and systems, they are shown as a matrix in figure D.4. The *Modular control system* function of section D.1 is omitted, as it is only another view of the *overall system* shown in the figure.

Even though most functions will only be implemented in one system, shown in **bold** in the figure, the function often needs to be taken into consideration with regard to other systems.

In two cases, the intersection between function and system is picked out as a separate area of effort. **Control of DT-VGT** is picked out because of the complexity of the subject, combined with its importance. **Overall I/O** is picked out because it contains some very interesting possibilities related to other projects.

The complete list of technical *Areas of effort* is as follows:

Systems Functions	Overall system	Embedded module	Central controller	DT-VGT	PA-10
Local Database	Architecture High level design	Low level design Implementation	Interface	Data representation Obtaining data	Data representation Obtaining data
Local control	Implementation– architecture	Implementation		Design Parameters Test	Design Parameters Test
Local configuration	Architecture High level design	Low level design Implementation	Interface	Integration	Integration
Local I/O	Architecture High level design	Low level design Implementation		Integration	Integration
Communication	Architecture High level design	HW integration SW low level design SW implementation	HW integration SW low level design SW implementation		
Motion planner interface	High level design		Low level design Implementation		
Central configuration	Architecture High level design		Low level design Implementation		
Central user interface	Architecture High level design		Low level design Implementation		

Figure D.4: Areas of effort in the systems/functions matrix

- Overall system
- Communication
- Generic I/O
- DT-VGT integration
- Control of DT-VGT
- PA-10 integration
- Embedded module
- Local configuration
- Local control
- Local I/O
- Local database
- Central controller
- Central configuration
- Motion planner interface
- Central User interface

Business

If the project should result in more than just a technological curiosity, it is vital to make an effort with respect to the various business aspects. The following areas of effort have been identified for this project.

- Intellectual property rights
- Attract investors
- Attract students
- Academic interaction
- Media cover
- Exhibitions

Organisational

In order to maintain a working environment for the project, it is necessary to pay attention to various areas related to the organisations and persons that interact with the project.

The areas of effort within the *Organisational* area have been identified as:

- Ph.D. thesis — Complying to University demands.
- Cooperation with internal partners (at MIP)
- Cooperation with local partners (Denmark)
- Corporation with other partners
- Supervise students at subprojects

Political

- Promote supporting activities

Management

- Project management for the project

D.5 Work Breakdown Structure

In order to manage the project effectively, the areas of effort are arranged in a hierarchical *Work Breakdown Structure* or WBS. Each element of the structure is assigned a unique label, a *WBS code* that identifies its level and placement within the structure. The WBS structure of the project is shown in figure D.5 and the WBS codes for the *areas of effort* — **AOE** — are listed in table D.1

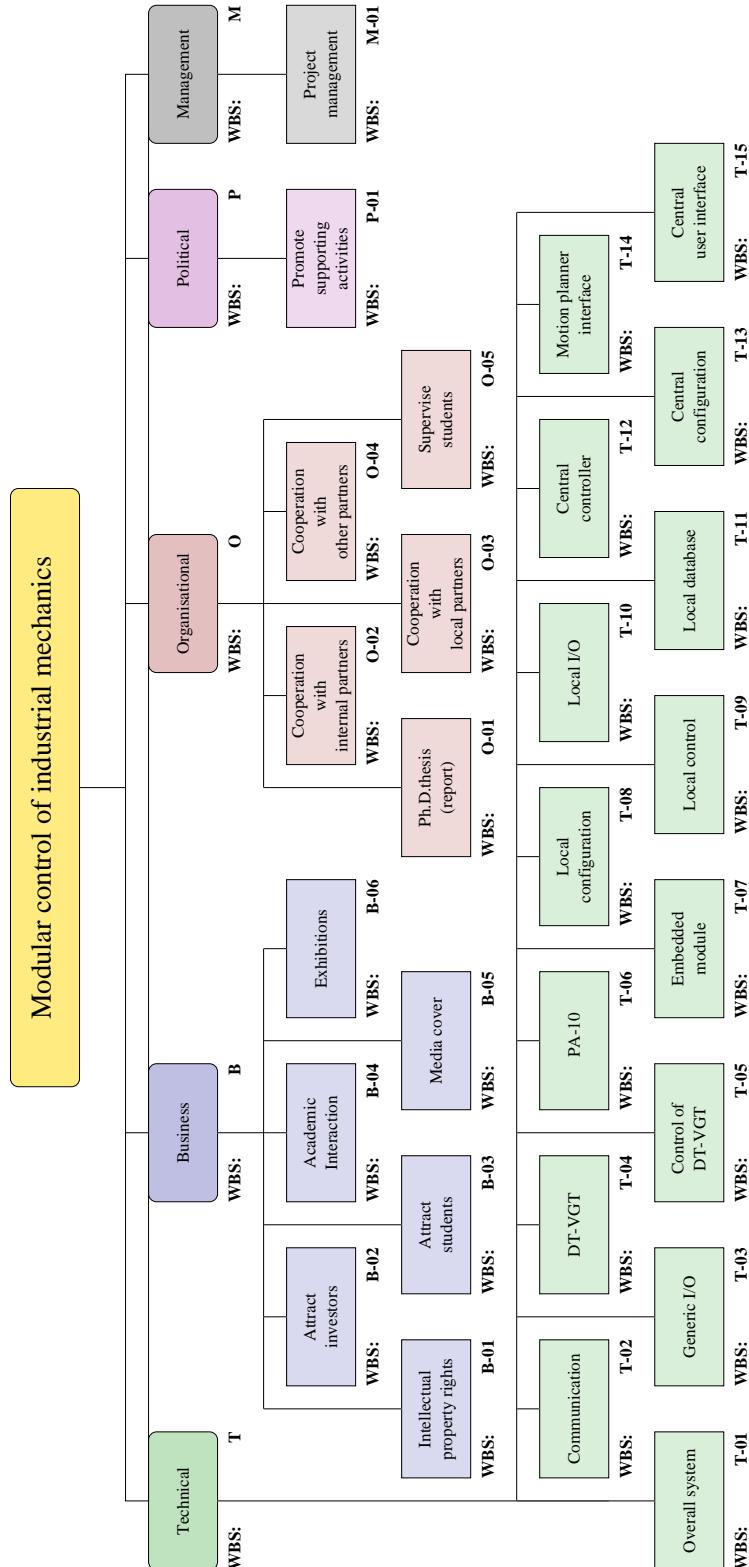


Figure D.5: Work breakdown structure of project (November 29. 2000)

WBS code	Area of effort (AOE)
T-01	Overall system
T-02	Communication
T-03	Generic I/O
T-04	DT-VGT integration
T-05	Control of DT-VGT
T-06	PA-10 integration
T-07	Embedded module
T-08	Local configuration
T-09	Local control
T-10	Local I/O
T-11	Local database
T-12	Central controller
T-13	Central configuration
T-14	Motion planner interface
T-15	Central User interface
B-01	Intellectual property rights
B-02	Attract investors
B-03	Attract students
B-04	Academic interaction
B-05	Media cover
B-06	Exhibitions
O-01	Ph.D. thesis — Complying to University demands
O-02	Cooperation with internal partners (at MIP)
O-03	Cooperation with local partners (Denmark)
O-04	Coorporation with other partners
O-05	Supervise students at subprojects
P-01	Promote supporting activities
M-01	Project management for the project

Table D.1: Primary WBS codes

Subdivision of AOE's

Each AOE can be broken down in a hierarchy, containing:

Activities are the atomic work units of the project.

They are denominated by the prefix **.a** followed by a 2-digit number identifying the activity within the AOE or cluster.

Milestones are items or events, that marks the end of an activity, a phase, or a similar transition.

They are denominated by the prefix **.m** followed by a 2-digit number, identifying the milestone within the AOE or cluster.

Clusters are used to subdivide an AOE into smaller units. A cluster can contain Activities, Milestones and clusters.

They are denominated by the prefix **.C** followed by a 2-digit number, identifying the cluster within the AOE or parent cluster.

Examples

1. Activities within AOE: T-01 are denominated:

T-01.a01 , T-01.a02 , ...

2. Milestones within AOE: B-03 are denominated:

B-03.m01 , B-03.m02 , ...

3. Clusters within AOE: O-05 are denominated:

O-05.C01 , O-05.C02 , ...

4. Activities within Cluster: .C04 of AOE: M-01 are denominated:

M-01.C04.a01 , M-01.C04.a02 , ...

D.6 Task descriptions

This section represents the overall project map, and has been maintained and updated as a separate document throughout the project. It describes each of the tasks performed in the course of the project, relating each task to the overall project tree.

This section will provide an overview of the tasks we have performed during the project, and it also provides a key to locate documents, design files, source code etc. in the project file hierarchy.

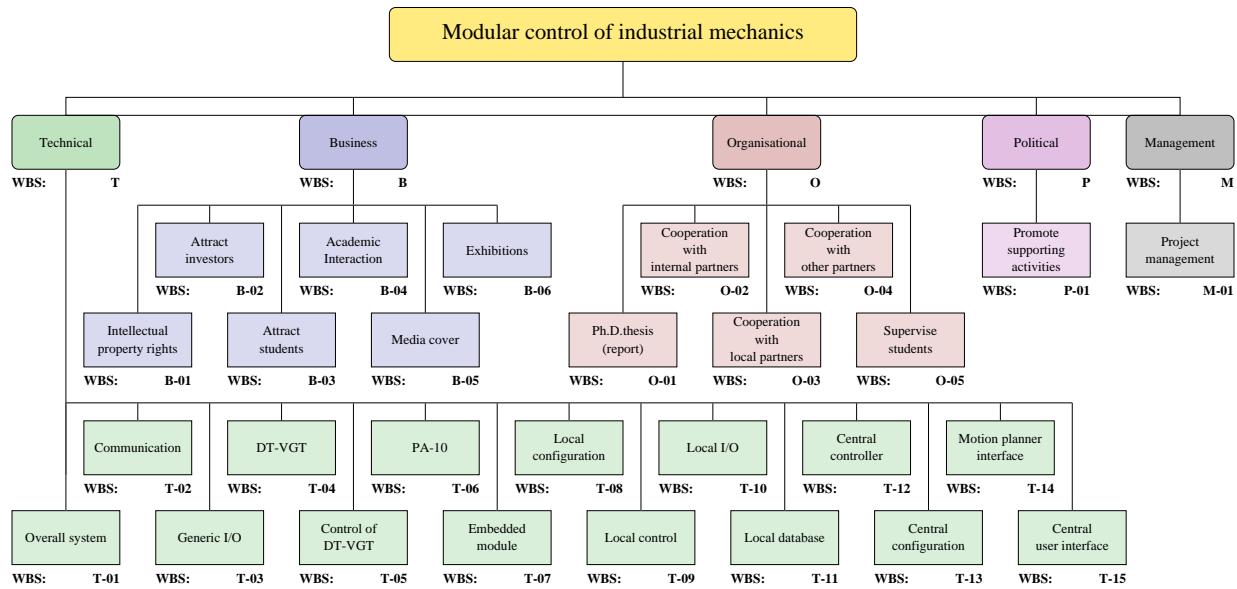


Figure D.6: Functional breakdown of project

T Technical

T-01 Overall system

This AOE is responsible for the overall system planning and integration, for the whole project.

T-02 Communication

This AOE is responsible for the communication system or systems linking the embedded modules together with each other, and the central controller.

T-02.C01 Evaluation of present technology

T-02.C02 Firewire between Orsys and Linux

T-02.C02.a01 Firewire support for Linux ix86: This activity is responsible for installing and adjusting the necessary hard- and software, to use IEEE-1394 Firewire on a standard PC running Linux.

T-02.C02.a02 Asynchronous data transfer: Developing the necessary application to transfer data between the Linux PC and the TMS320C32 computers via Firewire.

T-02.C02.a03 Isochronous data transfer:

T-02.C03 New API for Firewire interface

T-02.C03.a01 Investigation of Link Layer Interface: In this activity, the TSB13LV31 link layer controller IC is investigated from a programmers point of view.

T-02.C04 Integration of Firewire and control system

T-02.C04.a01 Transmitting a stream of data: In this activity, a Linux example (By Andreas Bombe) for transmitting an iso channel, is changed to work with Orsys example for receiving on an ISO channel.

T-02.C04.a02 Copy from ISO channel to I/O port: In this activity, an application for streaming the content of an ISO channel to an 8 bit digital output port is developed, as a test program for Linux as well as TMS320.

T-03 Generic I/O

This AOE is responsible for the generic I/O concept that is necessary to ensure that the embedded computer platforms can be connected to the widest possible array of sensor and actuator technology.

T-04 DT-VGT

This AOE is responsible for integrating the DT-VGT to the embedded modules.

T-04.C01 Switched mode valve amplifier

This cluster is responsible for the development of a high speed switched mode amplifier for the hydraulic valves. The cluster is covered by Jakob Lindeløv, who is developing the amplifier as his final engineering project, from the Engineering College of Copenhagen”.

T-04.C02 Follow up on valve amplifier

This cluster is responsible for the follow up on Jakob Lindeløv’s work on the switched mode amplifier, and it’s final integration with the DT-VGT’s

T-04.C02.a01 CPLD configuration: The CPLD controlling the amplifier transistors need to be reconfigured to meet the specifications for the amplifier.

T-04.C03 Generic I/O transition module

In order to interface hydraulics to the FPGA I/O board, a transition module is developed, with appropriate Analog and digital I/O components as well as 4 CAN-bus channels.

T-04.C03.a01 Module specifications:

T-04.C03.a02 Module design and implementation

T-04.C03.a03 Test of PWM outputs: The FPGA I/O module is configured with a set of PWM outputs compatible to the transition module.

T-04.C04 Integration with hydraulic test rig

In order to test the controller under simple and controlled conditions, it is integrated with a test rig for a single hydraulic actuator, before it is integrated with the DT-VGT itself.

T-04.C04.a01 Electronics rack: As the test rig will be used for several sub projects, the control electronics is integrated into a 19 inch rack to ensure stable and reliable operation for the duration of the project.

T-04.C04.a02 Test of rack: The test rig is submitted to varios simple test programs to verify that the control rack is functional.

T-04.C04.a03 Integration of valve amplifier: Buidling the amplifier into a box and integrating it with the test rig.

T-04.C04.a04 Misc. cabeling etc: Connecting the various systems of the test rig.

T-04.C04.a05 Integration of feedback valve: In order to evaluate the advantage of spool position feedback, a valve with spool feedback is integrated with the control system for the hydraulic test rig.

T-04.C05 Electronics enclosures

T-04.C05.a01 Evaluation of cooled enclosure: As the environmental temperature for the final prototypes can reach 60°C, some means of cooling must be applied in order to operate consumer grade electronics within sealed enclosures. In order to gain experience, we will evaluate a solution based on Peltier elements in an thermally isolated sealed metal enclosure.

T-04.C06 'First Move' demonstration

T-04.C06.a01 Integration of I/O subsystem: Adaptation and integration of VHDL modules to accomodate 3 actuator systems, using version 1 of our I/O mother-/daughter-board system.

T-04.C06.a02 Low level software architecture: Integration of modular Hardware with modular software.

T-04.C07 DT-VGT / Embedded controller

T-04.C07.a01 Integration of I/O subsystem: Adaptation and integration of VHDL modules to accomodate 3 actuator system version 1 of our I/O mother-/daughter-board system, fixing some of the problems encountered in the 'First move' demonstration.

T-04.C07.a02 Development of open-loop test system: In order to obtain the necessary data for modelling the hydraulic system, we must be able to perform a number of open-loop tests on the system.

T-04.C07.a03 Ad. hoc. feed forward based controller for paint demo: In order to demonstrate the VGT's ability to function as part in an experimental paint robot, we have enhanced our 'first-move' prototype with feed forward control, and matched the configuration to use 'Jacob Lindeløv's valve amplifiers.

T-05 Control of DT-VGT

This AOE is responsible for coming up with a useable control algorithm for the DT-VGT modules.

T-05.C01 Modelling and compensation for static characteristics

T-05.C01.a01 Modelling characteristics: Using results from various open-loop tests, we develop a model for the static characteristics of a single hydraulic actuator, mapping valve current to actuator speed.

T-06 PA-10

This AOE is responsible for integrating the PA-10 robot with the embedded modules.

T-07 Embedded module

This AOE is responsible for the design and implementation of the embedded computers.

T-07.C01 Orsys Software

The software from Orsys have to be altered slightly with respect to the use of makefiles, libraries, waitstates etc. in order to fit into the project.

T-07.C01.a01: Adaptation of stdio library.

T-07.C01.a02: Getting stdio to work without interrupts.

T-07.C01.a03: Merging the software for stdio and Firewire support.

T-07.C02 Low level software structure

T-07.C02.a01: Investigation of CPU overhead for Firewire communication.

T-07.C02.a02: Investigation of memory module concept In order to store data objects in a homogeneous way, a way to implement memory modules is investigated.

T-07.C03 Implementation of Mother board

The Mother board is the fusion of power supply, FPGA, ARC-net interface and similar static support systems into a single board, that can be used as basis board for all versions of our controller.

The detailed design and implementation is performed by Danny Kyrping.

T-08 Local configuration

This AOE is responsible for designing and implementing a concept for the individual states and state transitions of the embedded modules.

T-09 Local control

This AOE is responsible for designing and implementing a concept for handling the widest possible array of control algorithms, on the embedded modules.

T-10 Local I/O

This AOE is responsible for the design and implementation of an I/O system on the local embedded modules, based on the concepts from T-03.

T-10.C01 Microline/FPGA bus interface

T-10.C01.a01 Microline timing specification: The timing of the Microline bus is inadequately documented by Orsys. This activity is responsible for the necessary reverse engineering, to get adequate information about timing.

T-10.C02 FPGA I/O board for Microline

T-10.C02.a01 Testboard:

T-10.C02.a02 Prototype:

T-10.C02.a03 PROM emulator: we will design a small module that can be placed in the 8-pin PROM socket, and emulate the serial PROM. The configuration will be kept in a flash based reprogrammable device, programmable by an ordinary JTAG interface.

T-10.C03 VHDL I/O Framework

T-10.C03.a01 Generic I/O controller module: A VHDL module to interface between the Microline BUS, and a range of simple VHDL I/O modules.

T-10.C03.a02 8 bit digital output module: A VHDL module to output 8 bits.

T-10.C03.a03 2 bit hex output module: A VHDL module to output an 8 bit number on a dual 7-segment LED display.

T-10.C03.a04 32 bit hex output module: A VHDL module to output an 32 bit number multiplexed on a dual 7-segment LED display.

T-10.C03.a05 Top level design for Spartan II board: We adapt the generic I/O module design to fit the spartan II based I/O board.

T-10.C04 CAN interface

T-10.C04.a01 Simple test system: The FPGA is configured as a simple SPI interface between the DSP and CAN controller. The DSP uses the SPI interface to communicate with the CAN controller and thus a Temposonic II sensor.

T-10.C04.a02 Package handling by FPGA: The FPGA is configured so it can manage incoming CAN messages from a Temposonic III sensor, to relieve the DSP of this task.

T-10.C05 User interface

T-10.C05.a01 Playstation pad: A low cost ergonomic control pad for a playstation game console is connected to the FPGA, which is configured to communicate with the control pad.

T-10.C05.a02 no description:

T-10.C05.a03 no description:

T-10.C06 Spartan II based I/O board

As a replacement for the original XC40xx based I/O microline I/O board, we develop a board that combine: FPGA based I/O, Power supply and communication. The board will supply a Microline DSP board with power and RS-232 connectivity, while hosting a FPGA and an ARC-Net controller (COM 20022), connected to the microline bus.

The board is designed and implemented by Danny Kyrping.

T-11 Local database

This AOE is responsible for designing and implementing a concept for local storage of relevant kinematic and dynamic parameters on each distributed module.

T-12 Central controller

This AOE is responsible for the overall concept design and integration for the central controller.

T-13 Central configuration

This AOE is responsible for designing and implementing a concept for the global states and state transitions of the complete system.

T-14 Motion planner interface

This AOE is responsible for the interface between the set of distributed modules and the central motion planner.

T-15 Central user interface

This AOE is responsible for the application and user interface of the overall system.

B Business

B-01 Intellectual property rights

This AOE is responsible for handling possible IPR issues that may arise during the project.

B-02 Attract investors

To supplement the funding from MIP, it is essential to attract outside investors, such as the EEC. This AOE is responsible for these activities.

B-02.C01 ROBBOX

B-02.C02 GRAD

B-02.C03 Dockwelder

B-03 Attract students

It is quite important to attract students to the project, in order to get manpower and sustained interest. This AOE is responsible for attracting students to the project.

B-03.C01 Orientation meetings with students

At the end of each semester, MIP hosts a meeting with potential bachelor- and master-thesis students, to discuss the work areas of the institute and possible projects in these areas.

B-03.C01.a01 Preparation for talk at the fall 2000 meeting.

B-03.C01.a02 Preparation of a few detailed project descriptions as a follow up of the fall 2000 meetings.

B-03.C01.m01 The fall 2000 meeting.

B-03.C01.m02 The fall 2000 group meeting for computer-systems engineering.

B-04 Academic interaction

One of the traditional channels to attract outside interest, and thereby partners and investors, is by the traditional academic interaction through the publishing of papers and attendance of conferences etc.

B-04.C01 ISAM 2001

It is decided to participate with a paper and a conference talk, at the ISAM 2001 conference in Seoul, april 2001. This cluster contains all activities related to the participation in the conference.

B-04.C01.a01 Preparation of the first draft for the paper.

B-04.C01.m01 The editorial meeting for the draft paper.

B-04.C01.a02 Travel arrangements

B-04.C01.a03 Preparation of the talk in Seoul

B-04.C02 ISR 2002

B-04.C02.a01 Participation in writing the article.

B-04.C02.a02 Conference presentation.

B-04.C03 PKM

B-04.C03.a01 Writing the section about modular control.

B-05 Media cover

A less traditional, but potentially effective, way to attract interest in a project, is to attract the attention of the media.

This AOE is responsible for these efforts.

B-06 Exhibitions

The traditional way to get attention from relevant industrial partners, is to participate in industrial exhibitions.

This AOE is responsible for this angle.

O Organisational

O-01 Ph.D. thesis

Getting the Ph.D. degree is an important goal of this project. It is up to this AOE to ensure this outcome.

O-01.C01 Preparation of report

A total of 800 hours will be set aside for activities related to the writing of the final report.

O-01.C01.a01: First go at a report structure, derived from the WBS of the project.

O-01.C01.a02: Brain storm over subjects for report.

O-01.C02 Insitute labor

According to university rule, each Ph.D. student has to work 840 hours for the local institute.

At present, this quota has long been spend on various activities, such as:

- Teaching summer courses
- Technical support of other institute projects
- Descriptive work for funding applications

O-01.C03 Knowledge transfer

According to university rules, each Ph.D. student has to use 240 hours for knowledge transfer, which can be teaching or any other form of knowledge transfer, the Ph.D. board will agree to.

The quota has been met, by teaching, giving presentations, and education of other employees at MIP.

O-01.C04 Course participation

According to university rules, each Ph.D. student has to pass the equivalent of 0.5 year of high level university courses, relevant individual studying activities, or other studying activities, the Ph.D. board will agree to.

O-01.C04.a01 *Fuzzy control and neural networks* at DTU. Credit equivalent to 174 hours during the Fall-1999 semester.

O-01.C04.a02 *FPGA and VHDL self study.* Getting credit for studying the fine art of FPGA's and how to program them in VHDL. Estimated to give academic credit equivalent to 174 hours, but have taken a lot more time than that.

O-01.C04.a03 *Distributed systems* at DTU. Credit equivalent to 174 hours during the Fall-2000 semester.

O-01.C04.a04 *Project management* at DTU. Credit equivalent to 174 hours during the Fall-2000 semester, but the actual time consumed by this course is much higher.

O-01.C04.a05 *Written english communication* at MIP. Credit equivalent of 87 hours during a summer week 2000, but the workload is much lighter.

O-01.C04.a06 *FPGA's and VHDL for elastic I/O applications* Individual study activity in co-operation with Odense Engineering College. Evaluated by written report and oral project examination with internal censorship.

O-01.C05 Station exchange

All Ph.D. students must spend at least 3 consecutive weeks at an academic environment apart from their home institution.

I have fulfilled my quota by staying at DTU-IAU during November and December 2000.

O-02 Cooperation with internal partners

To keep the project running smoothly, it is important to keep the project rooted at SDU/MIP, by cooperating actively with individuals and groups.

O-02.C01 MIP technician

O-02.C01.a01 Description of work areas: The first step in acquiring the technician, is to describe his work areas and responsibilities. This will be done in cooperation with other MIP partners, specifically Søren Peder Jensen, and Henrik Hautop Lund.

O-02.C02 Employment at DockWelder

Due to the organisational and financial problems of the beginning of this project, I have been offered a temporary position working for DockWelder. This cluster contains the work required to realise the proposition.

O-02.C02.a01 Responsibility and work description: My past experiences with MIP and SDU, leads to a firm conviction that I will not accept any position at SDU without a thorough work description that is acceptable to myself, my employer, and the engineering union.

The formulation of a work and responsibility description is covered in this activity.

O-03 Cooperation with local partners

To ensure the necessary support from local partners such as Meganic ApS, AMROSE A/S, OSS and IOT, the project must pay attention to the cooperation with these partners.

O-03.C01 First move demonstration

The 'Applied mathematics or robotics' cooperate with AMROSE and Meganic in order to demonstrate the first move of the 'tulip' DT-VGT

O-03.C01.a01 Definition of common parameters: This activity coordinates definitions of coordinate systems, units, formats and protocols within the participants of 'First move'

O-03.C02 Controller reference model

In order to enter a dialogue with OSS, about controller development, it is important to have a clear definition of the controller concepts.

O-03.C02.a01 First definition of a layered reference model: A reference model will hardly be build in one day, this is the first shot.

O-04 Cooperation with other partners

As the project is becoming integrated with the EEC project DockWelder, cooperation with the foreign partners is an issue, which will be handled here.

O-04.C01 DockWelder Specifications

O-04.C01.a01 Presentation for sept 2001 meeting in Odense

O-05 Supervise students

O-05.C01 Benny Jørgensen

A Master thesis project about an AGV, involving FPGA based I/O

O-05.C01.a01 Development of FPGA based I/O system V.1.

P Political

P-01 Promoting supporting activities

P-01.C01 Computer systems engineering group

P-01.C01.a01 Presentation for manager: A short presentation of the group was prepared to allow the manager evaluate the potential.

P-01.C01.a02 Situation analysis: A situation analysis, illuminating the possibilities for a COSE group.

M Management

M-01 Project management

M-01.C01 Situation reports

M-01.C01.a01 Project history

M-01.C01.a02 Task description

M-01.C01.a03 Project partners

M-01.C01.a04 Environment

M-01.C01.a05 Uncertainties

M-01.C02 Resource analysis

M-01.C03 Organisation

M-01.C04 Management plan

M-01.C05 Methods and plans

M-01.C05.a01 Project structure:

M-01.C05.a02 Milestones:

M-01.C05.a03 Main plan:

M-01.C05.a04 Work plans:

M-01.C05.a05 Management plan:

M-01.C06 Attention areas

Bibliography

- [AD7856] Analog devices.*AD7856 data sheet.*
<http://www.farnell.com/datasheets/250.pdf> (feb 25. 2003)
- [Albus98] James S. Albus. A reference model architecture for intelligent systems design. *Intelligent Systems Division, Manufacturing Engineering Laboratory, National Institute of Standards and Technology*, 1998.
- [Anderson99] Don Anderson. *FireWire® System Architecture*. Addison-Wesley, One Jacob Way, Reading, Massachusetts 01867, USA, 2. edition, 1999. ISBN: 0-201-48535-4.
- [ARCNET] The ARCNET Trade Association. Arcnet trade association.
<http://www.arcnet.com> (mar 11. 2003)
- [CAN] Robert Bosch GmbH. Can homepage.
<http://www.can.bosch.com> (mar 10. 2003)
- [CMU-arm] Carnegie Mellon University The robotics institute. Reconfigurable modular manipulator.
<http://www-2.cs.cmu.edu/~paredis/rmms/> (dec 16. 2002)
- [COM20022] Standard Microsystems Corporation. *COM20022 Data sheet.*
<http://www.smsc.com/main/datasheets/20022.pdf> (mar 22. 2003)
- [Dalgaard01] Lars Dalgaard. Design, construction and control of a modular 2d-robot test platform. Master's thesis, The Maersk McKinney Møller Institute for Production Technology, University of Southern Denmark, Odense University, Campusvej 55, DK-5230 Odense M, Denmark, February 2001.
- [Dime] Nallatech. Dime a new modular standard with a fpga focus.
<http://www.nallatech.com/products/index.htm> (mar. 22. 2003)
- [DLR] Deutschen Zentrum für Luft-und Raumfahrt. Institute of robotics and mechatronics.
<http://www.robotic.dlr.de/mechatronics/lbr/> (dec 18. 2002)

- [EC-JRC] EC-JRC. The european commision joint research center (ec-jrc).
<http://www.jrc.it> (dec 6. 2002)
- [GENERIS] ERXA Software Solutions. Generis description.
<http://www.erxa.it/Eng/GENERIS/description.html> (dec 6. 2002)
- [Handel-C] Celoxica. Handel-c, language overview.
<http://www.celoxica.com/tech/handel-c/info.asp#> (feb 24. 2003)
- [IEEE-1394] IEEE Computer Society. Ieee standard for a high performance serial bus. Technical standard, Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017, USA, 1996. IEEE std 1394-1995.
- [Ifeachor93] Emmanuel C. Ifeachor and Barrie W. Jervi. *Digital Signal Processing, A Practical Approach*. Addison-Wesley Longman Limited, Edinburgh Gate, Harlow, Essex CM20 2JE, England, 1993. ISBN 0-201-54413-X.
- [Knudsen01] Jack Knudsen & Knud Hjørlund. Udvikling af modulært i/o system. Master's thesis, Ingeniørhøjskolen Odense Teknikum, Niels Bohrs Alle, DK-5230 Odense M, 2001.
- [Kyrping-A] Danny Kyrping. Documentation for embedded controller mother board.
 PCB layout, Schematics etc. : WBS: T-07.C03
- [Kyrping-B] Danny Kyrping. Documentation of dt-vgt daughterboard.
 Documentation for this project: WBS: T-04.C03.a02
- [Lindeløv01] Jakob Lindeløv. Design & implementering af intelligent effekttrin til hydraulikventil. Master's thesis, Ingeniørhøjskolen København, EIT-sektoren, Lautrupvang 15, DK-2750 Ballerup, Denmark, 2001.
- [Linux1394] The IEEE 1394 for Linux project group. IEEE 1394 for linux.
<http://www.linux1394.org> (mar 11. 2003)
- [Lundstrøm01] Mads Lundstrøm. Implementation af reeltidsnetværk til robotstyring, og styring af industri-robot. Bachelor thesis, The Maersk McKinney-Møller Institute for Production Technology, University of Southern Denmark, Odense University, Campusvej 55, DK-5230 Odense M, Denmark, 2001.
- [MCP2510] Microchip. *MCP2510 — Stand alone CAN controller with SPI^R Interface*, 1999.
<ftp://www.microchip.com/Download/lit/pline/analog/anicategcan/devices/mcp2510/21291c.pdf> (may 14. 2001)
- [Mezz] Mezzanines International industry group. Mezzanines international.
<http://www.mezzanines.org> (mar. 22. 2003).
- [Nanyang] Nanyang Technological University Robotics Research Center. Modular reconfigurable robots.
<http://155.69.254.10/users/risc/www/mod-intro.html> (dec 16. 2002).

- [Nicholas94] M. Nicholas. *Managing business and engineering projects*. Prentice Hall, Upper Saddle River, New Jersey, 1994. ISBN: 0-13-551854-7.
- [Nielsen02] Michael Bøllingtoft Nielsen. Realtids kommunikation mellem linix-pc og robotcontroller ved hjælp af tokenbus-netværk. Bachelor thesis, The Maersk McKinney-Møller Institute for Production Technology, University of Southern Denmark, Odense University, Campusvej 55, DK-5230 Odense M, Denmark, May 2002.
- [Nyquist] Nyquist Industrial control. Nyquist, about nyquist.
<http://www.nyquist.com> (mar 10. 2003)
- [Olsen00] Carsten Olsen. Modelling and control of a double tripod parallel robot. Master's thesis, The Maersk McKinney-Møller Institute for Production Technology, University of Southern Denmark, Odense University, Campusvej 55, DK-5230 Odense M, Denmark, November 2000.
- [OROCOS] The OROCOS project. Orococos (open robot control software).
<http://www.orocos.org> (mar. 30. 2003).
- [Orsys-A] Orsys Orth System GmbH, Am Stadtgraben 1 88677 Markdorf Germany. *Users guide microline® C32CPU*, 1.0 edition, 1998.
- [Orsys-B] Orsys Orth System GmbH, Am Stadtgraben 1 88677 Markdorf Germany. *Users guide microline® SC1394-1995 High Performance Serial Bus Communication Board*, 04 edition, 1998.
- [OSI] Multiple authors. Basic reference model for open systems interconnection. Technical report, The international standards organisation (ISO).
- [Pakpong01] Pakpong Jantapremjit & David Austin. Design of a modular self-reconfigurable robot. *Proceedings of Australian Conference on Robotics & Automation (ACRA), robot design and application*, November 2001.
<http://robot.anu.edu.au/david/publications/pa01b.pdf> (mar. 31. 2003)
- [PC/104] The PC/104 Consortium. PC/104 embedded-pc modules.
<http://www.pc104.org> (mar 22. 2003)
- [PCISIG] PCI-SIG PCI industry organisation. Pci-sig homepage.
<http://www.pcisig.com> (mar. 22. 2003)
- [Petersen01] H. G. Petersen K. Gregersen and M. L. Petersen. Distributed motion planning for modular robots. *SPIE proceeding volume 4571, Sensor Fusion and Decentralized Control in Robotic Systems IV*, 2001.
- [PICMG] Industrial Computer Manufacturers Group. Picmg.
<http://www.picmg.org> (mar. 22. 2003)

- [Powercube] Amtech Robotics. Powercube: Rotary and linear actuators.
<http://www.amtec-robotics.com/index2.html> (dec 16. 2002).
- [Riis98] Jens O. Riis Hans Mikkelsen. *Grundbog i projektledelse*. Forlaget Promet, Sømandshvile Park 7, DK-2960 Rungsted, 1998. ISBN: 87-89477-14-6.
- [SERCOS] Interessengemeinschaft SERCOS Interface e.V. Sercos interface.
<http://www.sercos.org> (mar 10. 2003)
- [Sørensen-A] Anders Stengaard Sørensen. Documentation of improvements to valve amplifier.
Documentation for this project: WBS: T-04.C02 (/Latex/amplifier.ps)
- [Sørensen-B] Anders Stengaard Sørensen. Documentation of microline bus timing.
Documentation for this project: WBS: T-10.C01.a01
- [Sørensen-C] Anders Stengaard Sørensen. Elastic i/o systems using fpga technologt and vhdl.
Supporting report with origin in this project: WBS: O-01.C04.a06
- [Sørensen-D] Anders Stengaard Sørensen. Embedded node source code for dt-vgt demonstra-tion.
Documentation for this project: WBS: T-04.C07.a02 (/TMS)
- [Sørensen-E] Anders Stengaard Sørensen. Modelling and control of hydraulic actuator.
Documentation for this project: WBS: T-05.C01.a01 (/Latex/main.ps).
- [Sørensen-F] Anders Stengaard Sørensen. Specification of DT-VGT FPGA configuration (using xilinx Foundation ISE)
Documentation for this project: WBS: T-04.C07.a01 (/VHDL)
- [Sørensen-G] Anders Stengaard Sørensen. Temposinic III interface.
Documentation for this project: WBS: T-10.C04 (/Latex/canbus.ps)
- [TMS320C32] Texas Instruments. *SPRS027C: Documentation of TMS320C32 DSP*, 1996.
<http://www-s.ti.com/sc/pssheets/sprs027c/sprs027c.pdf> (feb 10. 2001)
- [USB] USB Implementers Forum. Usb universal serial bus.
<http://www.usb.org> (mar 10. 2003)
- [VITA] VMEbus International Trade Association (VITA). Vita - open systems.
<http://www.vita.com> (mar. 22. 2003)
- [Wishbone] OpenCores. *WISHBONE, Revision B.3 Specification*, 2002.
http://www.opencores.org/wishbone/doc/specs/wbspec_b3.pdf (okt 3. 2002)
- [WorldFIP] WorldFIP. Worldfip fieldbus.
<http://www.worldfip.org> (mar 10. 2003)

- [XC2S] Xilinx Inc. *Spartan II 2.5V FPGA Family, Introduction and ordering information*, 2001.
http://direct.xilinx.com/bvdocs/publications/ds001_1.pdf (okt 3. 2002)
- [XC4000] Xilinx Inc. *XC4000E and XC4000X series Field Programmable Gate Arrays, product specification*, 1999.
<http://direct.xilinx.com/bvdocs/publications/4000.pdf> (okt 3. 2002)