



TAMPERE UNIVERSITY OF TECHNOLOGY

*Degree Programme in Machine Automation*

XIAOCHEN YU

VIRTUAL REALITY ENVIRONMENT FOR DEVELOPMENT  
AND SIMULATION OF MOBILE MACHINE

Master of Science Thesis

Examiner: Professor Asko Ellman  
Examiner and topic approved  
in the Mechanics and  
Design Department Council  
Meeting on 12 January 2011

# ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Master's Degree Program in Machine Automation

**YU, XIAOCHEN:** Virtual Reality Environment for Development and Simulation of Mobile Machine

Master of Science Thesis, 55 pages, 1 Appendix pages, and 31 figures.

January 2011

Major: Mechantronics

Examiner: Professor Asko Ellman

Keywords: Virtual Reality, Real-Time, SimMechanics, Simulation, xPC Target, UDP, Mobile Machine, Log loader.

This work was carried out at the Mechanics and Design Department of Tampere University of Technology. The primary purpose was to develop a real-time environment for a mobile machine. The thesis describes the process of the integration of a real-time simulating environment with Matlab. A mobile machine – loader equipment is modeled with SimMechanics and this model is built for running on the xPC Target which is a real time environment provided by Matlab. UDP were used to send and obtain the data, like position, velocity, acceleration and angle, between the host PC and the target PC. Finally this system is visualized in a Platform which is developed by Laboratory of Virtual Technology in TUT. The idea of this project is to develop a virtual reality environment by using Matlab in real-time. Also new ideas for improving the simulator and directions of further research were obtained with the help of this research.

## **PREFACE**

This master's thesis was carried out at the Department of the Mechanics and Design, Tampere University of Technology, Finland. I thank professor Asko Ellman for giving me an opportunity to participate in an interesting research project. I also thank him for his very big help, comments and appropriate suggestions. My special thanks to all of colleagues in our office for continuous support during this work. I also wish to thank my parents, my husband and my friends who supported me during my studies.

Xiaochen Yu

Tampere, Finland, December, 2010

# CONTENTS

	ABBREVIATIONS	VI
1	INTRODUCTION	1
1.1	Background	1
1.2	Overview on The System	3
1.3	Target of The Project	6
1.4	Organization of The Thesis	7
2	MODELING THE MECHANICAL SYSTEM IN SIMULINK WITH SIMMECHANICS TOOLBOX	8
2.1	Introduction	9
2.2	Functionality of the Toolbox	10
	2.2.1 Simulink Versus SimMechanics	10
	2.2.2 Physical Modeling Blocks	12
	2.2.3 Types of Analysis	12
2.3	Mathematical Aspects	13
	2.3.1 Modeling Multi-body Systems	13
	2.3.2 Equations of Motion for Multi-body Systems	14
	2.3.3 Sum up of The Mathematical Aspect in SimMechanics	16
2.4	Main steps of Modeling a Mobile Machine	16
	2.4.1 Creating a SimMechanics Model	17
	2.4.2 Connecting SimMechanics Blocks	17
	2.4.3 Interfacing SimMechanics Blocks to Simulink Blocks	17
	2.4.4 Setting SimMechanics Block Properties	18
	2.4.5 Creating SimMechanics Subsystems	19
	2.4.6 Creating Custom SimMechanics Blocks with Masks	20
3	SIMULATING THE SYSTEM IN THE TARGET PC WITH XPC TARGET	21

3.1	Overview	21
3.2	Hardware and Software Set	22
3.3	The host PC	23
3.3.1	The Real-Time Workshop	23
3.3.2	S-function in Simulink	24
3.4	Downloading A Model to the Target PC	25
4	COMMUNICATION AND VISUALIZATION THE MOBILE MACHINE BY UDP AND MEC VISUAL	30
4.1	Communication	30
4.1.1	Introduction of UDP	30
4.1.2	UDP Vs TCP	30
4.1.3	Reason for choosing UDP	32
4.1.4	Note on UDP Communication	32
4.2	Visualization	33
5	IMPLEMENTING THE VIRTUAL REALITY SYSTEM AND DISCUSSING THE RESULT	36
5.1	Model Building in SimMechanics	36
5.2	Booting The Target PC	38
5.3	Downloading The Model Into The Target PC	39
5.4	Simulating in Real-time	40
5.5	Communication	42
5.6	Visualization	44
5.7	Discussion	45
6	CONCLUSIONS AND FUTURE ENHANCEMENT	47
6.1	Conclusions	47
6.2	Suggestions for The Future Work	49
	REFERENCES	52
	APPENDIX	54

## ABBREVIATIONS

3D	3-Dimensional
CAD	Computer Aided Design
CAVE	Automatic Virtual Environment
DAE	Differential Algebraic Equation
DOF	Degrees of Freedom
HMD	Head Mounted Display
MEC	Department of the Mechanics and Design
ODE	Ordinary Differential Equation
PC	Personal Computer
TLC	Target Language Compiler
TUT	Tampere University of Technology
UDP	User Datagram Protocol
VR	Virtual Reality
VRML	Virtual Reality Modeling Language

# 1. INTRODUCTION

## 1.1 Background

Virtual Reality (VR) Environment is generally a Computer Generated environment that makes the user think that he/she is in the real environment. One may also experience a virtual reality by simply imagining it, like Alice in Wonderland, but we will focus on computer generated virtual realities for this thesis. [1]

Virtual Reality is an ideal training and visualization medium. It is ideal for the training of operators that perform tasks in dangerous or hazardous environments. The trainee may practice the procedure in virtual reality first, before graduating to reality-based training. The trainee may be exposed to life-threatening scenarios, under a safe and controlled environment.

Training simulators are typical VR equipments which have already been used extensively for several decades in teaching people to use some devices or machines without the need to use the actual system. The real world systems are either too dangerous or too expensive to be used by a person just learning how to use it. Simulated systems have ranged from simulations of commercial investment environments thru military strategy simulators to car driving and shuttle flying simulators. In the Figure 1.1 there is an example of a driving simulator. [2]



*Figure 1.1. A Driving Simulator (Carlton) [2]*

The definition of a real-time simulator is not an easy one. There are quite many definitions of what makes a system a real-time system, but a more clear definition is that of between a soft and a hard real-time system. A hard real-time system consists of several devices, which includes head mounted displays or flat screen (for example LCD) displays, joy sticks, computers, the fixed base plate attached to the floor with all six degrees of freedom and so on. The choice of what kind of equipment is used depends on the actual system simulated, the reality level desired, the level of immersion needed and of the cost of the system. A soft real-time system, through modeling and simulating the system in the host pc and target pc, to get real-time signals, then use those signals to visualize the system on the flat screen. In this project, we focus on establishing the soft real-time environment with SimMechanics and xPC Target in Matlab on hardware – two PCs. That means the real-time environment which I built in this thesis includes not only soft real-time systems, but also hard real-time systems. Developing the real-time environment is base on two PCs (hard) and established the model of the mobile machine with the Toolbox of Matlab in real-time (soft).





*Figure 1.2. Host PC and Target PC in This Project*

## 1.2 Overview on the system

This thesis is based on a research work in the Department of Mechanics and Design of the University of Tampere. The software tool Matlab is the primary software used in the department for communication and control because of its advantages. Therefore it is a very interesting research topic to develop an interface to drive the mobile machine by using Matlab.

The primary purpose of this work was to implement a model of mobile machine for real-time use. The thesis contains some theory and basic concepts on the Virtual Reality environment and the mobile machine, and some general theory of SimMechanics and xPC Target. Also the idea behind the model of the system is shown. The results of the study and the discussion of these are thereafter presented.

There are two possibilities to implement this interface. The simplest approach is to use the basic software SimMechanics in Matlab solely. This solution is only possible simulation mode. The drawbacks of this method are possibly the inaccurate timing (not real-time).

Because of this drawback, the second possibility comes with the idea of implementing a real-time interface for the mobile machine. The xPC Target from the Matlab family provides a small and simple real time kernel to realize this interface. It means we could implement this mobile machine driving interface in real-time or not. If only the SimMechanics is used, it may be not a real-time interface. However if we want to get a real-time interface, the simple way is the integration of both two toolboxes – SimMechanics and xPC Target in Matlab.

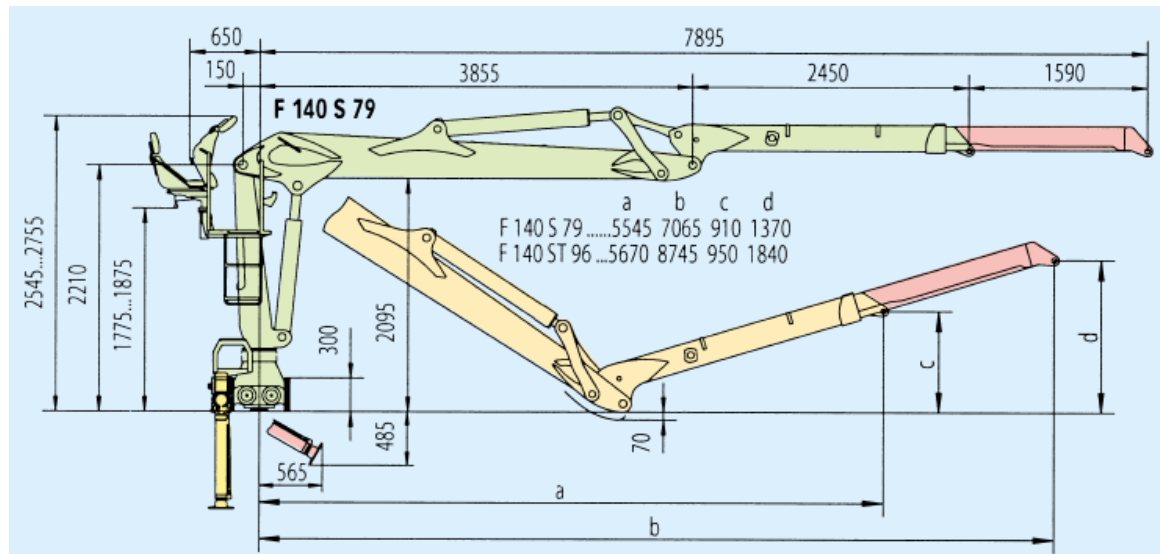
In order to find the real-time behavior of the interface, a real-time control application for the mobile machine by using Matlab will be included in the thesis. This real-time control application will drive a mobile machine moving along a specifically track by giving input signals. The track of the machine can be a straight one or a curved one.

One of the most important tasks for this thesis is to find out the functionalities provided by the Matlab for a real-time control system. Matlab is a high level, standard and widely spread software. It is a high-performance language for technical computing. The SimMechanics, Real-Time Workshop and xPC Target from the Matlab family will be used in this thesis, as they are very powerful in building and simulating a real-time mobile machine system. The SimMechanics is a software package used to model, simulate, and analyze systems whose outputs change over time. The Real-Time Workshop is an extension of capabilities found in SimMechanics and Matlab to enable rapid prototyping of real-time software applications on a variety of systems. The xPC Target is a solution for prototyping, testing, and deploying real-time systems using standard PC hardware. All these products from the Matlab family are commonly used to realize a real-time control application whose behavior is changed over time. They are spread in almost every university, company and research center, so in this thesis they are used to develop this real-time environment. The final product of this thesis will be a SimMechanics block diagram running on the xPC Target.

The task of the xPC Target in this application is to design a mobile machine by constructing a SimMechanics block diagram. This block diagram will be uploaded to the target PC from the host PC. After the running of the application, the target PC will communicate with the host PC via UDP.

For the mobile machine, the Hydraulic Log Loader Loglift 140 S was chosen in this

application. This crane is an Optimal combination of hydraulics and mechanics. The model consists of a vertical frame and three boom sections. Hydraulic cylinders controlling the angle between the frame and the first boom and between the first boom and the second boom are modeled, the joint between the second boom and the third boom are a telescopic joint.



*Figure 1.3. The Hydraulic Log Loader Loglift 140 S [3]*

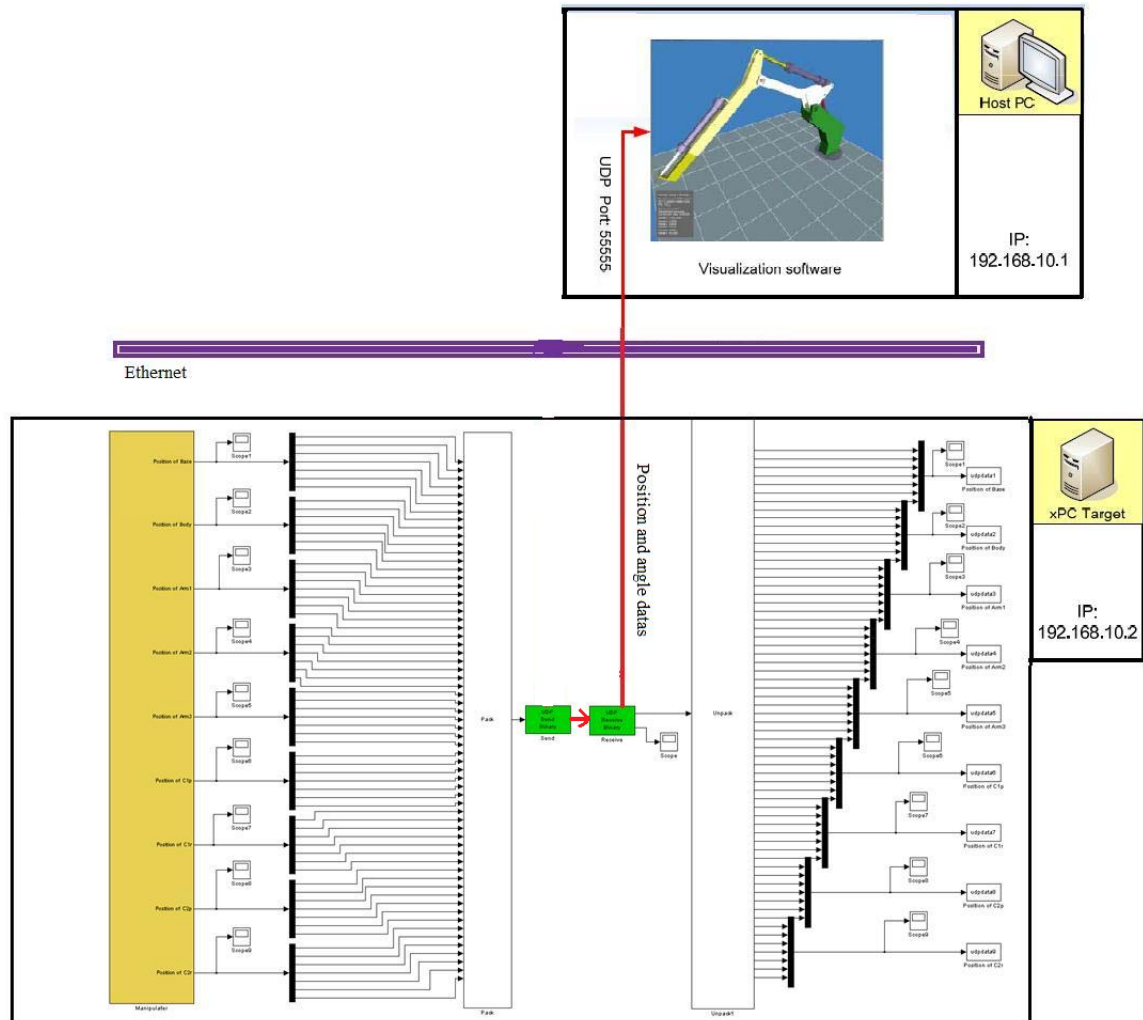
The implementation of this application is to build a communication block in a SimMechanics diagram. By using this SimMechanics block diagram the xPC Target can communicate with the target PC in real-time. Here the communication means the xPC Target will establish a connection between the host PC and the target PC, and through this connection the application running on the target PC can receive information from the host PC and send the position and rotation data of the log loader to the MEC visual which is a visualization software developed by the Department of the Mechanics and Design, Tampere University of Technology. The received information will be used to visualize the log loader.

This part can be divided into three steps:

1. The aim of the first step is to build a model of the mobile machine - the Hydraulic Log Loader Loglift 140 S with SimMechanics.
2. In the second step, download that model to the target PC and run it in the real-time with

## XPC Target

3. The task of the third step is to get the datas of real-time and visualize the dynamic mobile machine.



*Figure 1.4. Datas Transfer from The Target PC to The Host PC*

## 1.3 Target of the project

Target of the work was to integrate all the software into a functional real-time environment with a demonstration simulation of the Hydraulic Log Loader Loglift 140 S.

## 1.4 Organization of the thesis

The list of abbreviations is in front of this thesis. Appendix is the List of figures, and it is at the end of the thesis.

This thesis is broken up into two main parts: theory part and experiment part. The theory part according to how to integrate the Virtual Reality environment and the main steps taken in the realization of the real-time control system of a mobile machine. After this introduction chapter, the next five chapters detail the design process, the hardware, the software, the testing and the resulting problems.

Chapter 2 systematically presents the mathematical and software developments needed for efficient simulation of mechanical systems in the SimMechanics dynamic simulation environment and the main steps of the model building.

Chapter 3 starts with which hardware and software are used in this project, and reasons for them. The design issue includes the principle of the xPC Target and how to use it to simulate the mobile machine in real-time.

Chapter 4 opens with the communication and the visualization tools and also the theoretical background.

In chapter 5, lots of experiments are used to analyze the behavior of the VR system.

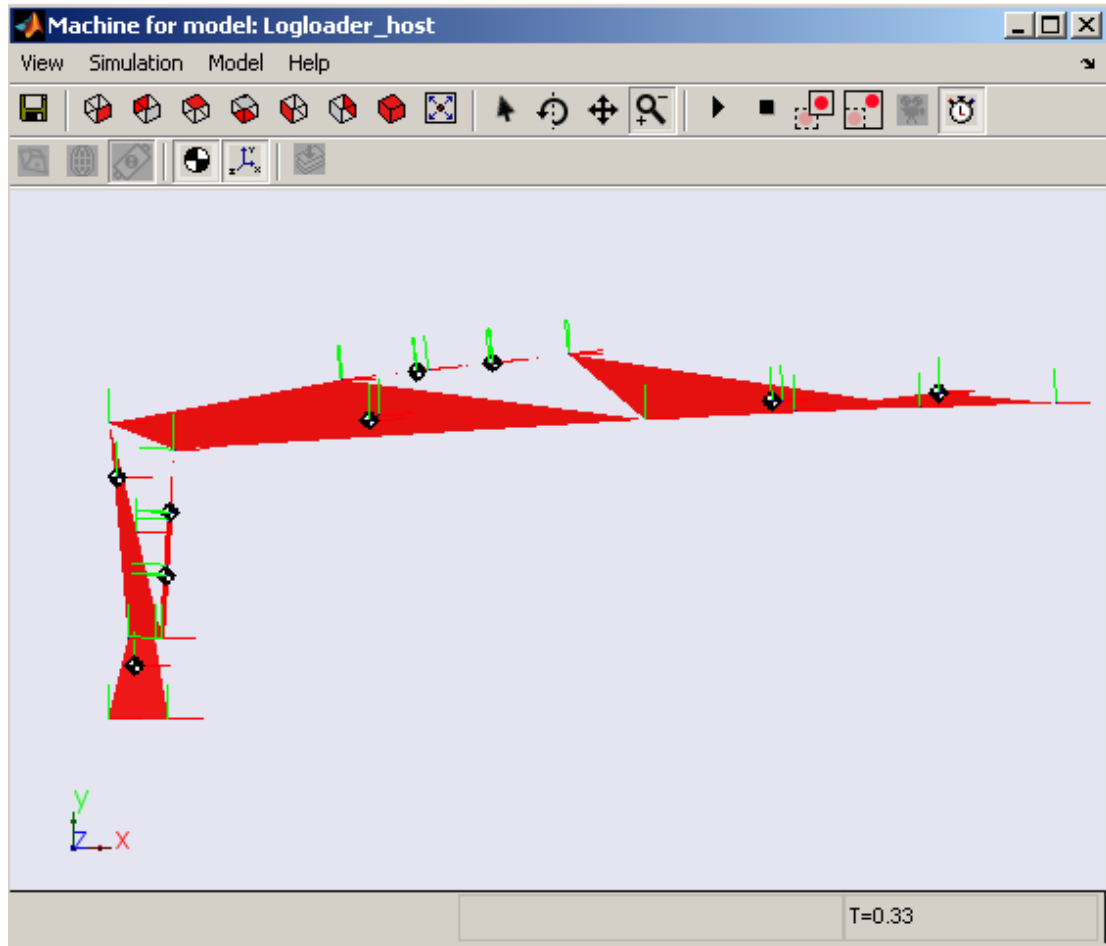
Chapter 6 summarizes the contributions of this thesis and poses suggestions and goals for future work.

## **2. MODELING THE MECHANICAL SYSTEM IN SIMULINK WITH SIMMECHANICS TOOLBOX**

SimMechanics extends Simulink with tools for modeling and simulating mechanical systems. SimMechanics simulates the motion of mechanical devices and generates mechanical performance measurements associated with this motion. It is integrated with MathWorks control design and code generation products, enabling us to design controllers and test them in real time with the model of the mechanical system. SimMechanics can be used for a variety of aerospace, defense, and automotive applications, such as the development of active suspension, antiroll-over, landing gear and control surface actuation systems.

SimMechanics software is a block diagram modeling environment for the engineering design and simulation of rigid body machines and their motions, using the standard Newtonian dynamics of forces and torques. With SimMechanics software, we can model and simulate mechanical systems with a suite of tools to specify bodies and their mass properties, their possible motions, kinematic constraints, and coordinate systems, and to initiate and measure body motions. Unlike other Simulink blocks, which represent mathematical operations or operate on signals, SimMechanics blocks represent physical components or relationships directly. We represent a mechanical system by a connected block diagram, and we can incorporate hierarchical subsystems.

The visualization tools of SimMechanics software display and animate simplified standard geometries of 3-D machines, before and during simulation. SimMechanics gives us access to all standard MATLAB graphics functions, and automatically creates three-dimensional animations of your mechanical system. With the Virtual Reality Toolbox (available separately) you can connect to a VRML environment and create presentation quality animations. Figure 2.1 shows a MATLAB visualization of the Hydraulic Log Loader with SimMechanics.



*Figure 2.1. A MATLAB Visualization of The Hydraulic Log Loader with SimMechanics*

## 2.1 Introduction

This chapter describes how to simulate the dynamics of multi-body systems with SimMechanics, a toolbox for the Matlab/Simulink environment. The novel Physical Modeling method is compared with traditional ways to represent mechanical systems in Matlab. It also systematically presents some interesting mathematical aspects of the implementation of SimMechanics.

Simulating the dynamics of multi-body systems is a common problem in engineering and science. Various programs are available for these tasks which are either symbolical computation programs to derive and solve the dynamical equations of motion, or numerical programs which compute the dynamics on the basis of a 3D-CAD model or by means of a

more abstract representation, e.g. a block diagram.

Mechanical systems are represented by connected block diagrams. Unlike normal Simulink blocks, which represent mathematical operations, or operate on signals, Physical Modeling blocks represent physical components, and geometric and kinematic relationships directly. This is not only more intuitive, it also saves the time and effort to derive the equations of motion.

SimMechanics models, however, can be interfaced seamlessly with ordinary Simulink block diagrams. This enables the user to design e.g. the mechanical and the control system in one common environment. Various analysis modes and advanced visualization tools make the simulation of complex dynamical systems possible even for users with a limited background in mechanics.

## **2.2 Functionality of the Toolbox**

This section provides an overview about SimMechanics. The block set is described briefly, as well as the different analysis modes and visualization options.

### **2.2.1 Simulink VS. SimMechanics**

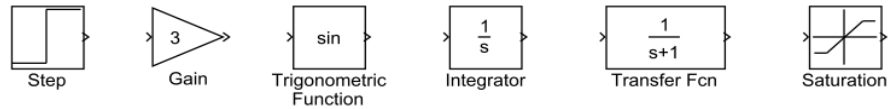
#### **Simulink**

Simulink software models, simulates, and analyzes dynamic systems. It enables us to pose a question about a system, model the system, and see what happens.

With Simulink, we can easily build models from scratch, or modify existing models to meet our needs. Simulink supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Systems can also be multirate — having different parts that are sampled or updated at different rates.

Simulink models represent the mathematics (algebraic/differential equations) of a dynamical system. We must derive equations of motion by ourself. Figure 2.2 shows the models of Simulink.





**Figure 2.2.** *The Simulink Models*

Thousands of scientists and engineers around the world use Simulink to model and solve real problems in a variety of industries, including:

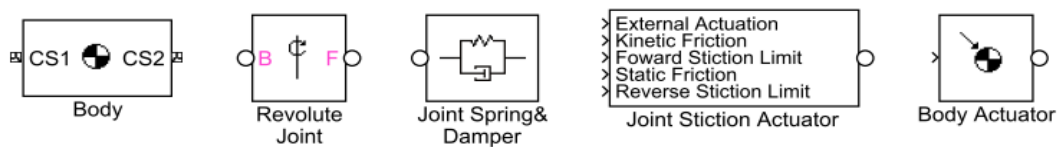
- Aerospace and Defense
- Automotive
- Communications
- Electronics and Signal Processing
- Medical Instrumentation

### **SimMechanics**

SimMechanics software is a block diagram modeling environment for the engineering design and simulation of rigid body machines and their motions, using the standard Newtonian dynamics of forces and torques.

With SimMechanics software, we can model and simulate mechanical systems with a suite of tools to specify bodies and their mass properties, their possible motions, kinematic constraints, and coordinate systems, and to initiate and measure body motions. Representing a mechanical system by a connected block diagram, like other Simulink models, and we can incorporate hierarchical subsystems.

SimMechanics machines represent the physical structure of a mechanical system, and equations of motion are derived by itself. Figure 2.3 shows the models of SimMechanics.



**Figure 2.3.** *The SimMechanics Models*

### **2.2.2 Physical Modeling Blocks**

As already mentioned, the SimMechanics blocks do not directly model mathematical functions but have a definite physical (here: mechanical) meaning. The block set consists of block libraries for bodies, joints, sensors and actuators, constraints and drivers, and force elements. Besides simple standard blocks there are some blocks with advanced functionality available, which facilitate the modeling of complex systems enormously. An example is the Joint Stiction Actuator with event handling for locking and unlocking of the joint. Modeling such a component in traditional ways can become quite difficult. Other features are Disassembled Joints for closed loop systems. If a machine with a closed topology is modeled with such joints, the user does not need to calculate valid initial states of the system by solving systems of nonlinear equations.

Standard Simulink blocks have distinct input and output ports. The connections between those blocks are called signal lines, and represent inputs to and outputs from the mathematical functions. Due to Newton's third law of action and reaction, this concept is not sensible for mechanical systems. If a body A acts on a body B with a force  $F$ , B also acts on A with a force  $F$ , so that there is no definite direction of the signal flow. Special connection lines, anchored at both ends to a connector port have been introduced with this toolbox. Unlike signal lines, they cannot be branched, nor can they be connected to standard blocks. To do the latter, SimMechanics provides Sensor and Actuator blocks. They are the interface to standard Simulink models. Actuator blocks transform input signals in motions, forces or torques. Sensor blocks do the opposite. They transform mechanical variables into signals.

### **2.2.3 Types of Analysis**

By simulating the SimMechanics model, we can impose kinematic constraints, apply forces and torques, and measure the resulting motions and forces. We can also develop and test motion-driving actuators, such as electric motors, ball screws, hydraulic cylinders, and engines.

SimMechanics supports five motion analysis modes:

**Forward Dynamics**—Assigns driving forces and torques to the motion-driving actuators and calculates the resulting motions of the entire system

**Inverse Dynamics and Kinematics**—Determines the forces and torques that must be exerted by the actuators to produce user specified motions

**Trimming**—Identifies the steady-state equilibrium points to be used for linearization and system analysis

**Linearization** — Extracts a linear model that predicts the system’ s response to perturbations in driving forces, joint and constraint configurations, and initial conditions Collectively, these modes of analysis enable you to test mechanical performance, select proper actuation systems, and develop optimal controls.

Generally it is necessary to build a separate model for each type of analysis, because of the different mechanical variables which need to be specified. In Forward Dynamics mode, the initial conditions for positions, velocities, and accelerations, as well as all forces acting on the system are required to find the solution. To use the Inverse Dynamics or the Kinematics mode, the model must specify completely the positions, velocities, and accelerations of the system’s independent degrees of freedom.

However, analysing the model is not the key point in this thesis. In this project we should focus on how to integrate all the software into a real-time environment for a mobile machine.

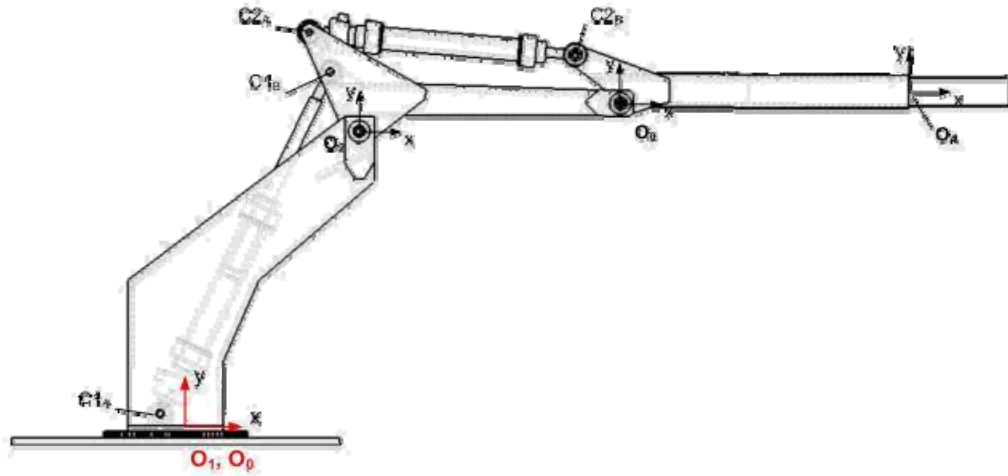
## **2.3 Mathematical Aspects**

In this section it is examined how SimMechanics works. Some interesting mathematical aspects are addressed, but the underlying algorithms are not described.

### ***2.3.1 Modeling multi-body systems***

To simulating the mobile machine in real-time, we must build the model in Simlink with SimMechanics first. Actually this model which just built with Simmechanics does not run in real-time as mentioned in chapter 1.2, paragraph 3. Then, if we want to simulate the mobile machine in real-time, another toolbox – xPC Target must be used. In this chapter, I just show the first step which is how to build the model of the mobile machine in SimMechanics and some mathematical aspect behind this toolbox.

We have to be familiar with the concept of a multi-body system as an abstract collection of bodies whose relative motions are constrained by means of joints and other, perhaps more complicated, constraints. We also have to be familiar with the representation of a multi-body system by means of an abstract graph. The bodies are placed in direct correspondence with the nodes of the graph, and the constraints (where pairs of bodies interact) are represented by means of edges. Figure 2.4 shows the abstract graph of the Hydraulic Log Loader Loglift 140 S. This one is more easy to understand and create model in SimMechanics than Figure 1.3 because of the simplification of the joint  $C2_B$  and  $O_3$ .



*Figure 2.4. The Abstract Graph of The Hydraulic Log Loader Loglift 140 S*

### 2.3.2 Equations of Motion for Multi-body Systems

There are numerous derivations of the governing equations of motion of a multi-body system [4, 5, 6, 7, 8]. Various protagonists adhere quite strongly to certain formulations in preference to others. But, in all cases, the equations of motion for multibody systems can always be written as:

$$\dot{\mathbf{q}} = \tilde{\mathbf{H}}(\mathbf{q})\mathbf{v} \quad (\text{kinematical eq.}) \quad (1)$$

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbf{f}(t, \mathbf{q}, \mathbf{v}) + \tilde{\mathbf{H}}^T(\mathbf{q})\mathbf{G}^T(\mathbf{q}, t)\boldsymbol{\lambda} \quad (\text{dynamical eq.}) \quad (2)$$

$$\mathbf{g}(\mathbf{q}, t) = 0 \quad (\text{constraint eq.}) \quad (3)$$

Equation (1) expresses the kinematic relationship between the derivatives of the

configuration variables  $\mathbf{q}$  and the velocity variables  $\mathbf{v}$ . In the most simple cases  $\tilde{\mathbf{H}}$  is the identity matrix. The equation (2) is the motion equation with the positive definite mass matrix  $\mathbf{M}$ , the acceleration  $\dot{\mathbf{v}}$ , the contribution of the centrifugal, Coriolis, and external forces  $\mathbf{f}$ , and the contribution of reaction forces due to kinematical constraints which is expressed by the last term on the right side. Finally, equation (3) represents kinematical constraints between the configuration variables  $\mathbf{q}$ .

The main problem arising from equations (1)–(3) is that they form an index-3 differential algebraic equation (DAE) (The index of a DAE is one plus the number of differentiations of the constraints that are needed in order to be able to eliminate the Lagrange multiplier  $\lambda$ .) because of the constraints in equation (3). Currently, Simulink is designed to simulate systems described by ODEs and a restricted class of index-1 DAEs, so the multibody dynamics problem is not solvable directly. [9]

The structure of the equations of motion depends largely on the choice of coordinates. Many commercial software packages for multi-body dynamics use the formulation in absolute coordinates. In this approach, each body is assigned 6 degrees of freedom first. Then, depending on the interaction of bodies due to joints, etc. suitable constraint equations are formed. This results in a large number of configuration variables and relatively simple constraint equations, but also in a sparse mass matrix  $\mathbf{M}$ . This sparseness and the uniformity in which the equations of motions derived are exploited by the software. However, due to the large number of constraint equations, this strategy is not suitable for SimMechanics. It uses relative coordinates. In this approach, a body is initially given zero degrees of freedom. They are “added” by connecting joints to the body. Therefore, far fewer configuration variables and constraint equations are required. Acyclic systems can even be simulated without forming any constraint equations. The drawback of this approach is the dense mass matrix  $\mathbf{M}$ , which now contains the constraints implicitly, and the more complex constraint equations (if any).

The form of the equations of motion and the structure of their matrices depend on the type of coordinates that define the configuration of the system. Ideally it should be possible to choose a minimal set of independent coordinates, equal in number to the global degrees of freedom exhibited by the mechanical system. This eliminates the need for constraint equations. Unfortunately, for anything other than the most trivial systems, it is not possible to obtain a global chart parameterizing the manifold defined by the equation 3. The problem

of solving the system defined by equation 3 (an index-3 DAE) remains. Simulink, not at present designed for DAE analysis, has difficulty solving such systems.

### ***2.3.3 Sum up of The Mathematical Aspect in SimMechanics***

In this chapter we have addressed some of the issues that arise when introducing mechanical simulation capability into an existing ODE-based simulation environment. We addressed the issue of computational efficiency by demonstrating how a relative coordinate formulation can be implemented using recursive computations to produce efficient algorithms. The problems of Lagrange multiplier  $\lambda$  determination and preventing numerical drift were also addressed. A number of strategies were outlined. Each has different properties which make it applicable in different situations. [10]

SimMechanics is a commercial tool for simulating mechanical systems. It is an extension of Matlab Simulink software. Its key features are ease of use and integrability to existing Simulink block diagrams. Mechanical systems (linear and nonlinear) can be modeled with SimMechanics blocks that can be connected to Simulink blocks. CAD models can also be directly translated to function blocks by using separate software. SimMechanics also offers Matlabs powerful mathematical tools, such as integration and optimization. Furthermore other Matlab extensions, such as the Real-Time Toolkit can be integrated into the development environment. The drawback of this software for mobile machine is that there is no built-in collision detection, but the user must take care of this. The system also offers automatic C-code generation. The following section will introduce the main steps of modeling a mobile machine in SimMechanics.

## **2.4 Main Steps of Modeling a Mobile Machine**

SimMechanics provides tools for building mechanical models that include bodies, joints, coordinate systems, and constraintsjoints, coordinate systems, and constraints. We can connect SimMechanics blocks with Simulink blocks to include non-mechanical, multi-domain effects in your mechanical models. We can save models that combine Simulink and SimMechanics blocks as subsystems for reuse in many applications.

### ***2.4.1 Creating a SimMechanics Model***

Creating a SimMechanics model is as much same as creating any other Simulink model. First, we must create a Simulink model window, then drag instances of Simulink blocks from the Simulink block library into the window and draw lines to interconnect the blocks.

The SimMechanics block library provides the following blocks specifically for modeling machines:

- Body blocks that represent a machine's components and the machine's immobile environment, or ground
- Joint blocks that represent the degrees of freedom of one body relative to another body or to a point on ground
- Constraint and Driver blocks that restrict motions of or impose motions on bodies relative to one another
- Initial condition blocks that specify the initial state of the machine
- Actuator blocks that specify forces or motions applied to joints and bodies
- Sensor blocks that output the forces and motions of joints and blocks

We can use blocks from other Simulink libraries in SimMechanics models. For example, we can connect the output of SimMechanics Sensor blocks to Scope blocks from the Simulink Sinks library to display the forces and motions of the model's bodies and joints. Similarly, we can connect blocks from the Simulink Sources library to SimMechanics Driver blocks to specify relative motions of the machine's bodies.

### ***2.4.2 Connecting SimMechanics Blocks***

In general, the way to connect SimMechanics blocks is as same as the way to connect other Simulink blocks: by drawing lines between them. Significant differences exist, however, between connecting standard Simulink blocks and connecting SimMechanics blocks.

### ***2.4.3 Interfacing SimMechanics Blocks to Simulink Blocks***

SimMechanics Actuator blocks contain standard Simulink input ports. Thus, we can connect standard Simulink blocks to a SimMechanics model via Actuator blocks. Similarly, SimMechanics Sensor blocks contain output ports. Thus, we can connect a SimMechanics model to Simulink blocks via Sensor blocks.

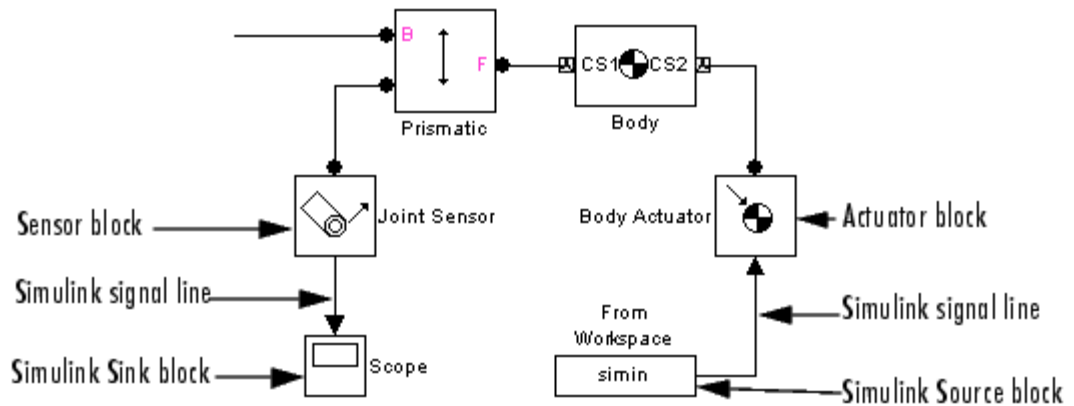


Figure 2.5. The Interface between SimMechanics Blocks and Simulink Blocks [11]

### 2.4.4 Setting SimMechanics Block Properties

The block parameters must be set via the block dialog boxes. We can open the dialogs by double-clicking the block, or by right-clicking the block and selecting Open block.

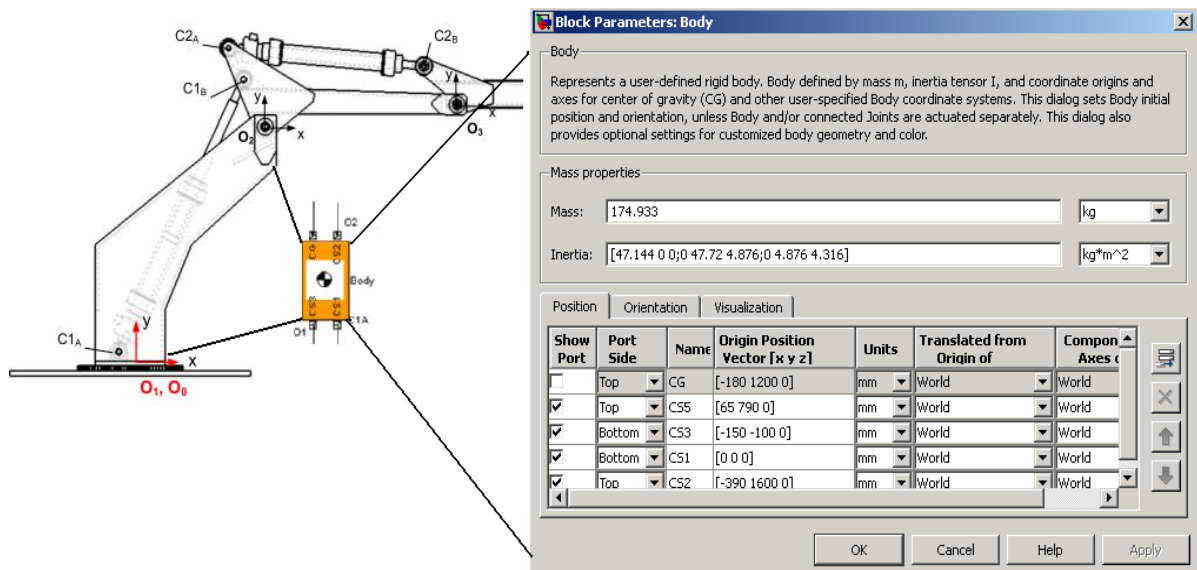


Figure 2.6. The Block Parameters of The Crane

### 2.4.5 Creating SimMechanics Subsystems



Large, complex block diagram models are often hard to analyze. Enclosing functionally related groups of blocks in subsystems alleviates this difficulty and facilitates reuse of block groups in different models.

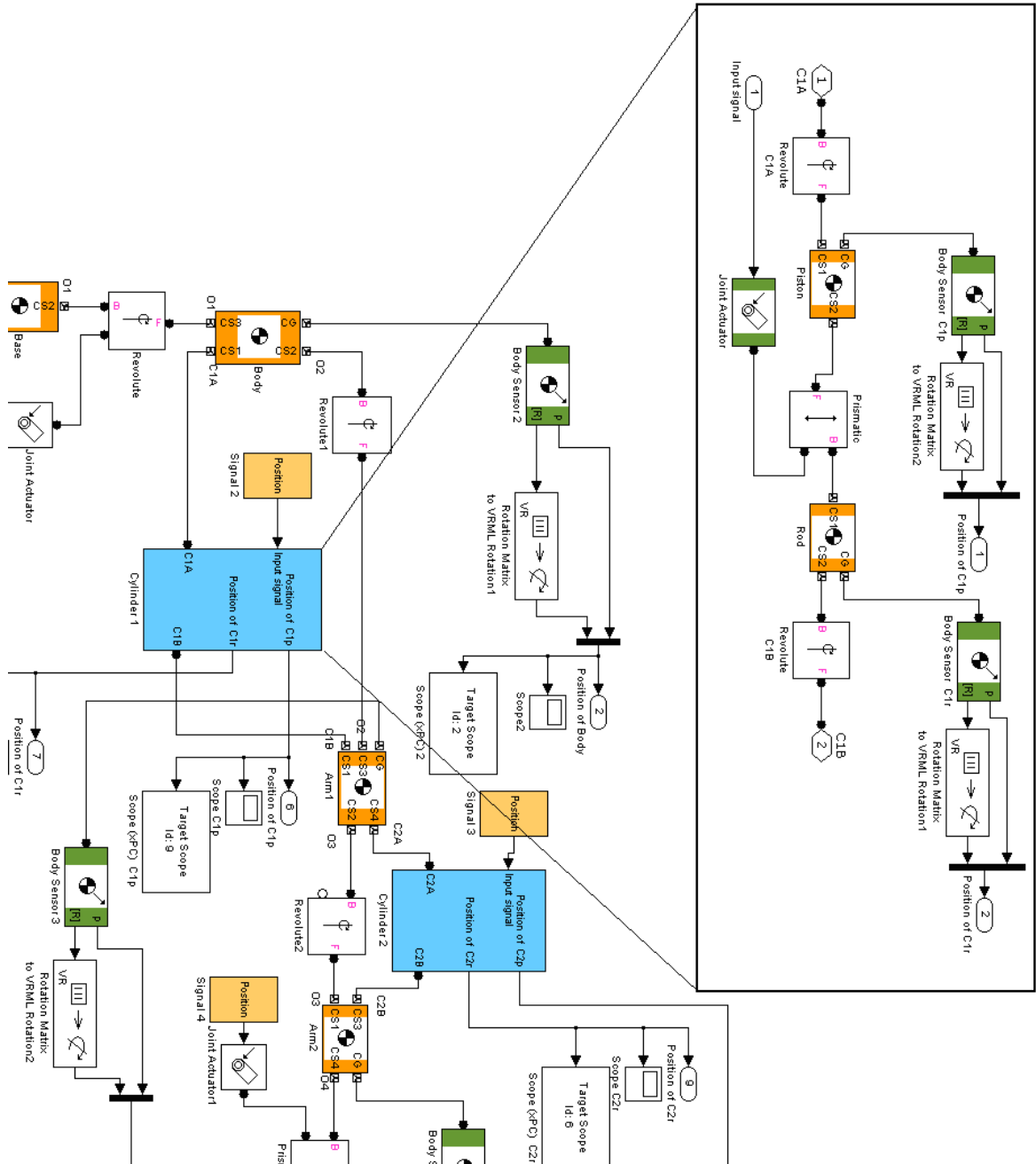


Figure 2.7. The Subsystem of The Cylinder of The Hydraulic Log Loader Model

We can create subsystems containing SimMechanics blocks that we can connect to other SimMechanics blocks.

### ***2.4.6 Creating Custom SimMechanics Blocks with Masks***

I also can create my own SimMechanics blocks from subsystems, for example, a spring-loaded Joint block or a sphere Body block. To do this, create a block diagram that implements the functionality of the custom block, enclose the diagram as a subsystem, and add a mask (i.e., user interface) to the subsystem. To facilitate sharing the custom blocks across models or with other users, create a Simulink block library and add these masked subsystem blocks to the library. The Simulink User's Guide explains how to create custom blocks with masks.

## **3. SIMULATING THE SYSTEM IN THE TARGET PC WITH XPC TARGET**

### **3.1 Overview**

The title of this thesis shows this is a real-time environment application. Currently, most control applications are based on PC control and the use of the Windows operating system. In this application the Windows operating system will only be used on the host PC. The reason is that it is not an exact real-time operating system. For real-time requirements of this application, the xPC Target from the MATLAB is chosen to realize this control task. It is a solution for prototyping, testing, and deploying real-time systems using standard PC hardware and is an environment that uses a target PC, separate from a host PC, for running real-time applications. [12] Changing parameters in the target application while it is running in real time, and checking the results by viewing signal data, are two important prototyping tasks. xPC Target includes a command-line and graphical user interfaces to complete these tasks. This environment includes a host PC and a target PC. The host PC is a development platform which has Visual C, MATLAB SimMechanics and Real-Time Workshop. The target PC is just a normal PC which can be booted with an xPC boot disk. xPC Target does not require DOS, Windows, Linux, or any another operating system on the target PC. Instead, the target PC is booted with a boot disk that includes the highly optimized xPC Target kernel. It is created on the host PC by setting up the xPC Target environment properties for example the xPC Target kernel specific for either serial or network communication. To create a target application, a SimMechanics model will be created at first. (introduced in chapter 2) xPC Target then uses the SimMechanics model, Real-Time Workshop, and a third-party compiler to create the target application on the host PC. Real-Time Workshop provides the utilities to convert the SimMechanics models into C code and then with a third-party C/C++ compiler, compile the code into a real-time executable. This executable is then converted to an image suitable for xPC Target and uploaded to the target PC.

The working procedure of the xPC Target is first to develop a real-time application on the host PC. This application is built by creating a MATLAB model file which uses the Simulink and SimMechanics libraries. The model file will be compiled as a real-time executable and then uploaded to the target PC. The target PC will provide a real-time environment for the application. The application can be started or stopped from the target PC or the host PC. Both the target and host PC can control the application when it is running. For example the parameters of the application can be changed from the host or target PC during the run time. The xPC Target hardware requires a host PC and a target PC. The target PC only needs to have the I/O boards supported by xPC Target and a boot disk. More details about the software on the host PC and target PC and how they work together will be introduced in the following sections.

### 3.2 Hardware and Software Set

In this application environment I use my desktop computer as a host PC with MATLAB, Simulink, SimMechanics and xPC Target software, to create a model using SimMechanics blocks. After creating my model, I can run simulations in non-real time in host PC.

xPC Target software lets me add I/O blocks to my model and then use the host PC with Real-Time Workshop, Real-Time Workshop Embedded Coder, and a C/C++ compiler to create executable code. The executable code is downloaded from the host PC to the target PC running the xPC Target real-time kernel. After downloading the executable code, I can run and test my target application in real time.

- Hardware requirements — The xPC Target software requires a host PC, target PC, and, for I/O, the target PC must also have I/O boards supported by the xPC Target product. However, the target PC can be a desktop PC, industrial PC, PC/104, PC/104+, or Compact PCI computer.
- Software requirements — The xPC Target software requires either a Microsoft® Visual C/C++ compiler (Version 6.0, 7.1, 8.0, or 9.0) or an Open Watcom C/C++ compiler (Version 1.7). In addition, the xPC Target software requires MATLAB, Simulink, and Real-Time Workshop software.

The software I used during the whole project is listed below.

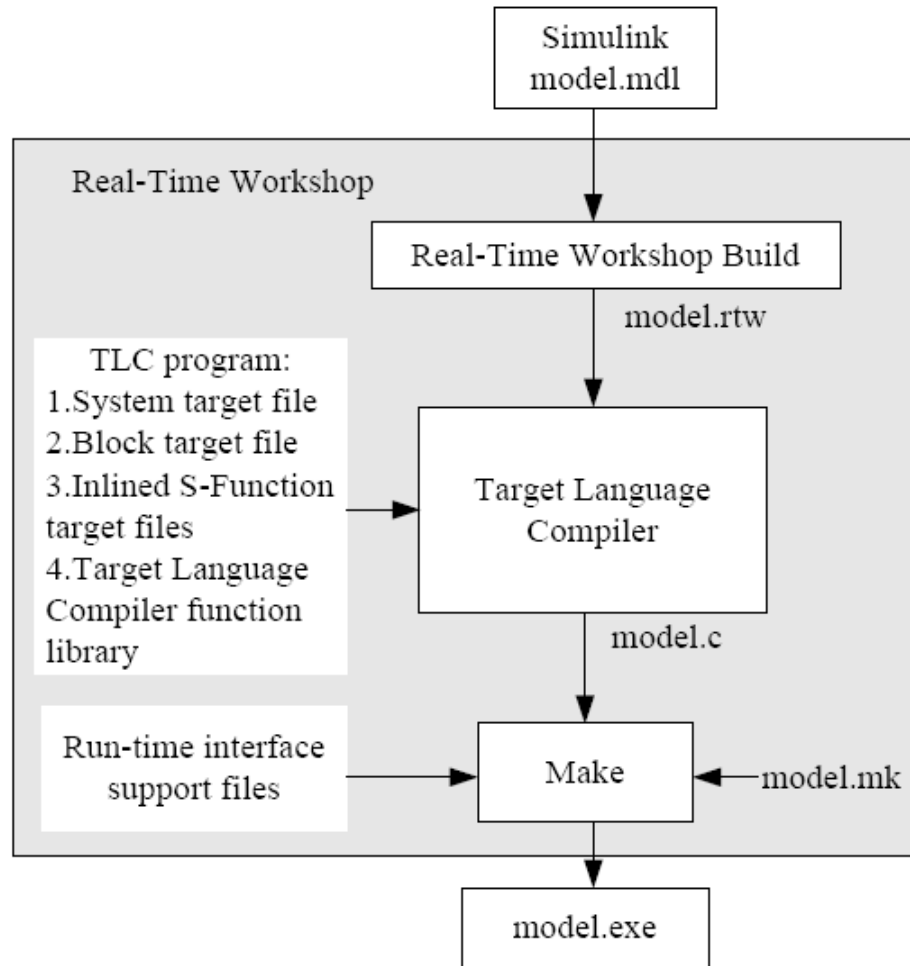
- Windows xp operating system
- MATLAB, Version 7.9.0.529 (R2009b)
- VISUAL C
- Real-Time Workshop for MATLAB R2009b
- xPC Target. Toolbox for MATLAB R2009b

### **3.3 The Host PC**

The MATLAB, Simulink and C Compiler are common softwares used in every university, they will not be detailed in this thesis. The only thing need to be mentioned is the Real-Time Workshop and MATLAB S-Functions (system functions).

#### ***3.3.1 The Real-Time Workshop***

Real-Time Workshop is an extension of capabilities of Simulink and MATLAB that automatically generated, packages and compiles source code from Simulink models to create real-time software applications on a variety of systems. Real-Time Workshop provides the utilities to convert a Simulink models into C code and then, with a third-party C/C++ compiler, compile the code into a real-time executable. As shown in Figure (3.1) the main task of the Real-Time Workshop is to compile a Simulink model file to an executable file which will work on different targets. More details about the Real-Time Workshop can be found in [13].



*Figure 3.1. Real-Time Workshop Code Generation Process [12]*

With Real-Time Workshop, State flow Coder (optional) and a C compiler on the host computer, the executable code is created. The xPC Target requires the source code of every Simulink block to be written in "C". The Simulink scheme to be downloaded must not contain any call to M-files. Parameters can be passed to a S-function in various ways, by making initialization files, loading parameters into workspace, or have prior blocks giving the correct parameters. After creating the executable code, the target application can be run in real time on a second PC compatible system.

### **3.3.2 S-function in Simulink**

S-Functions (system-functions) provide a powerful mechanism for extending the capabilities of Simulink. The most common use of S-Functions is to create custom

Simulink blocks. It can be used for a variety of applications, including:

- Adding new general purpose blocks to Simulink
- Adding blocks that represent hardware device drivers
- Incorporating existing C code into a simulation
- Describing a system as a set of mathematical equations
- Using graphical animations

An advantage of using S-Functions is that it can be used to build a general purpose block that can be used many times in a model, varying parameters with each instance of the block. In this application the S-Functions work together with the Real-Time Workshop to solve various kinds of problems. These problems include:

- Extending the set of algorithms (blocks) provided by Simulink and Real-Time Workshop
- Interfacing existing (hand-written) C-code with Simulink and Real-Time Workshop
- Generating highly optimized C-code for embedded systems

The explanation of S-Function above means that the S-Function is a tool which allows the developer to create their own MATLAB tool block for their special applications. Simulink is a program that runs as a companion to MATLAB, these programs are developed and marketed by the MathWorks, Inc.

### **3.4 Downloading A Model to the Target PC**

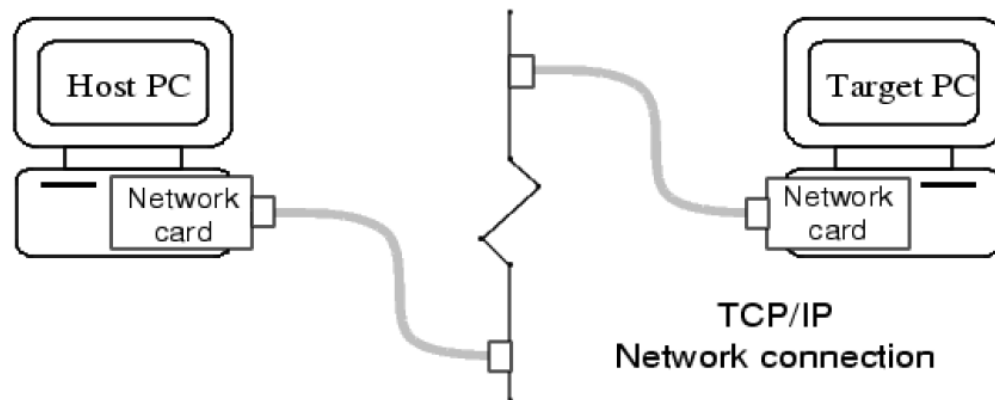
The target PC does not require any operating system, like Windows, UNIX or even DOS, but is instead booted with a CD. This CD contains the highly optimized xPC Target kernel, created on the host computer by typing the command "xpcsetup" in the common MATLAB prompt. The kernel makes the target ready for downloading of applications from the host. This application need to have source code written in C. After downloading the application, the kernel makes the target ready for execution, and it is ready for real-time use. The application is written into the RAM at designated places by the kernel.

For communication between the target and the host, there are two options, serial connection or network connection. Network connection is in general faster, up 10Mbit=s instead of

100kBaud=s. More important, it does not restrict the distance between host and target to the length of the serial cable.

In this section a simple introduction about how the xPC Target works will be given. The first step is to build a connection between the host PC and the target PC. There are two ways to connect them: one is the serial communication (e.g serial RS232) and the other is using network communication. Here the network communication is used to connect the target and the host PC because it has two advantages:

- Higher data throughput transfer up to 100 Mbit/second
- Longer distances between host and target computer



*Figure 3.2. Hardware Connection between Target PC and Host PC [12]*

Figure (3.2) shows the connection of the Host PC and Target PC. The second step is to create a Target Boot Disk. Figure (3.3) shows the configuration window of creating a Target Boot Disk. The third step is to boot the Target PC with the created Target Boot Disk. After successfully booting the Target PC a window like Figure (3.4) will appear which means the connection between the Host PC and Target PC are connected. The fourth step is to use a test program provided by MATLAB to test if a Simulink model file can be uploaded to the Target PC and run. This is done in the MATLAB Command Window, by typing "xpctest". MATLAB runs the test script and displays messages indicating the success or failure of a test.

```
### xPC Target v4.2 Test Suite
### Host-Target interface is: TCP/IP (Ethernet)
```



```
### Test 1, Ping target PC 'TargetPC1' using system ping: ... OK
### Test 2, Ping target PC 'TargetPC1' using xpctargetping: ... OK
### Test 3, Software reboot the target PC 'TargetPC1': ??? Error while
evaluating TimerFcn for timer 'timer-4'
```

TCP/IP Connect Error

```
..... OK
### Test 4, Build and download an xPC Target application using model
xpcosc to target PC 'TargetPC1': ... OK
### Test 5, Check host-target command communications with
'TargetPC1': ... OK
### Test 6, Download a pre-built xPC Target application to target PC
'TargetPC1': ... OK
### Test 7, Execute the xPC Target application for 0.2s: ... OK
### Test 8, Upload logged data and compare with simulation results: ...
OK
### Test Suite successfully finished.
```

If all of the tests succeed, that means it is ready to build and download a target application to the target PC.

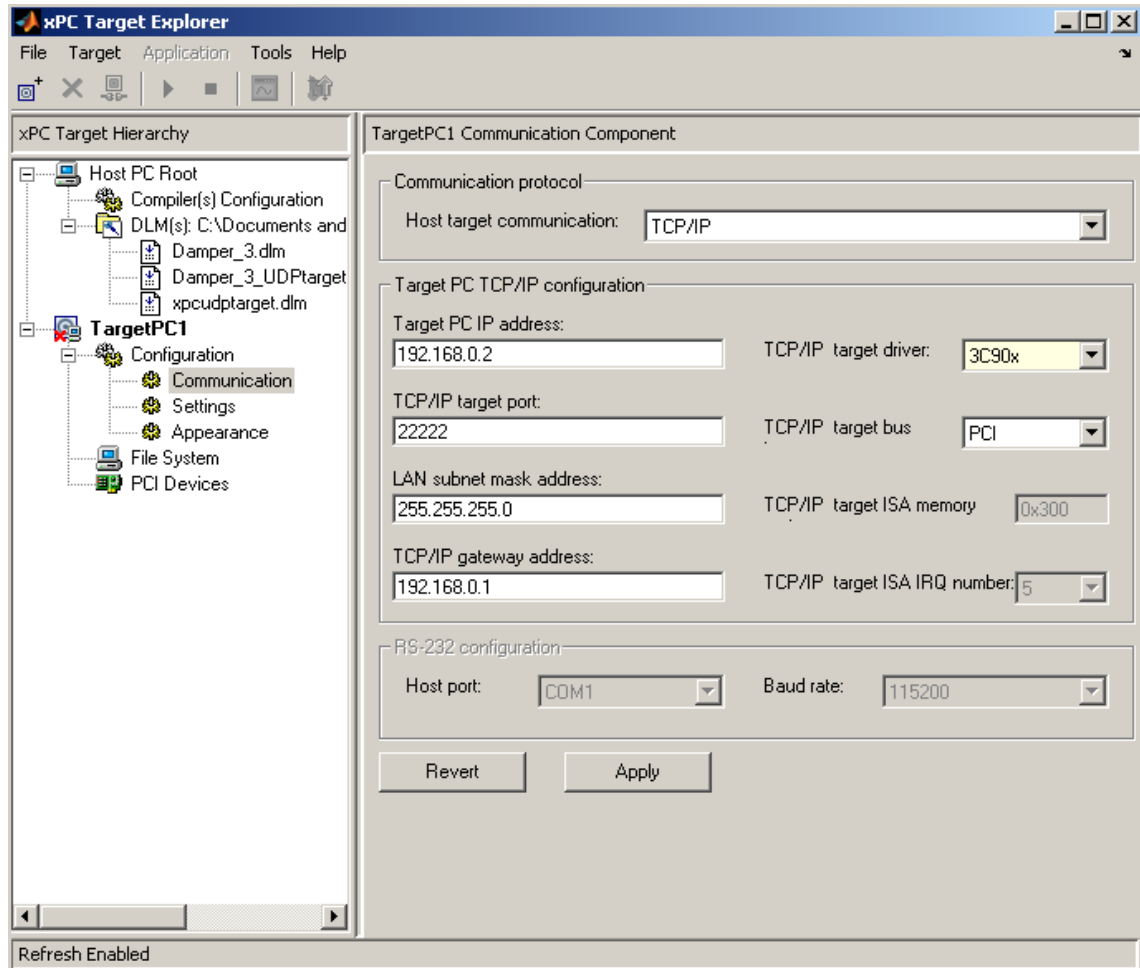


Figure 3.3. Configuration of Target Boot Disk

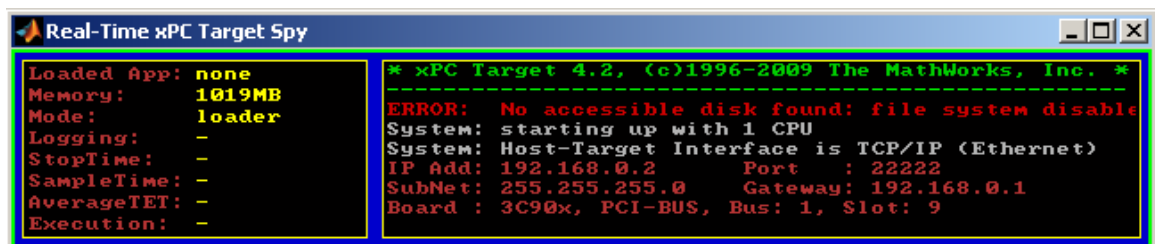
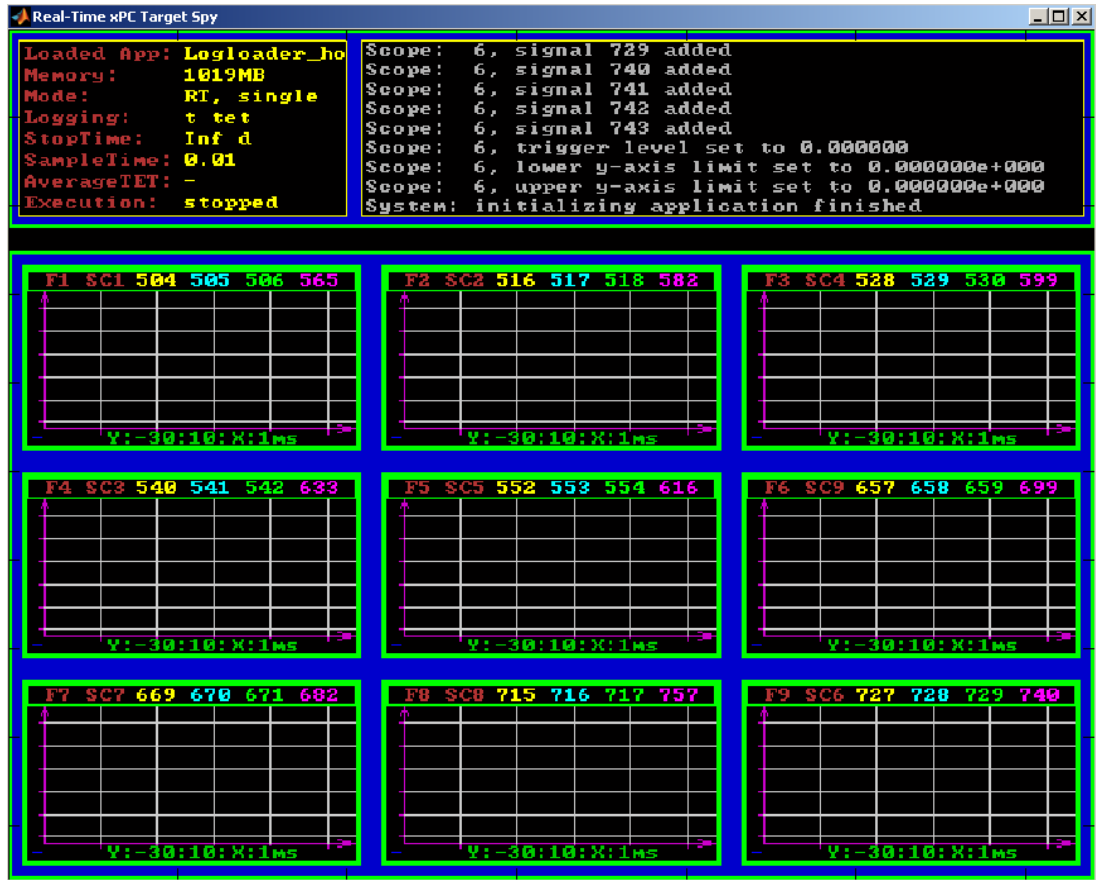


Figure 3.4. xPC Target Boots, The Kernel and Display Information on The Target PC



*Figure 3.5. Loading a MATLAB SimMechanics Model to The Target PC*

The above message shows that the xPC Target is successfully connected. As seen in Figure (3.5), the name of the model, the size of the model, the sample rate of the model and the status of the model are displayed in the xPC Target's information window. More information about the xPC Target can be found in [14] and also will be introduced in the following chapters during the implementation of this application.

## **4. COMMUNICATION AND VISUALIZATION THE MOBILE MACHINE BY UDP AND MEC VISUAL**

### **4.1 Communication**

The communication between the processes is performed via the computer network. It is therefore possible for the controller process reside on different computers.

#### ***4.1.1 Introduction of UDP***

Communication takes place using the User Datagram Protocol (UDP). UDP packets are sent by one process to a specific port on a specific computer. A process on the receiving computer listens to that port. Since one of the involved processes is a Matlab process in this case, and Matlab only handles Matlab arrays, special functions must be used to pack, send, receive and unpack packets containing Matlab arrays. Such functions have been implemented for numerical Matlab arrays. The UDP primitives are available in Matlab through a number of C-MEX functions. In Matlab, one, two, three or four arrays are packed into one UDP packet. Also, an arbitrary string is included in the packet to provide a mechanism for packet identification. UDP is not lossless like TCP, but in our setup, with a dedicated, full duplex, point-to-point 100 Mbits/s ethernet link, the loss is negligible.

The xPC Target software supports communication from the target PC to other systems or devices using User Datagram Protocol (UDP) packets. UDP is a transport protocol similar to TCP. However, unlike TCP, UDP provides a direct method to send and receive packets over an IP network. UDP uses this direct method at the expense of reliability by limiting error checking and recovery. To use UDP for the xPC Target system, be sure to create a TCP/IP boot disk and boot the target PC with that boot disk.

#### ***4.1.2 UDP Vs TCP***

The User Datagram Protocol (UDP) is a transport protocol layered on top of the Internet Protocol (IP) and is commonly known as UDP/IP. It is analogous to TCP/IP. A convenient way to present the details of UDP/IP is by comparison to TCP/IP as presented below:

- **Connection Versus Connectionless** — TCP is a connection based protocol, while UDP is a connectionless protocol. In TCP, the two ends of the communication link must be connected at all times during the communication. An application using UDP prepares a packet and sends it to the receiver's address without first checking to see if the receiver is ready to receive a packet. If the receiving end is not ready to receive a packet, the packet is lost.
- **Stream Versus Packet** — TCP is a stream-oriented protocol, while UDP is a packet-oriented protocol. This means that TCP is considered to be a long stream of data that is transmitted from one end to the other with another long stream of data flowing in the other direction. The TCP/IP stack is responsible for breaking the stream of data into packets and sending those packets while the stack at the other end is responsible for reassembling the packets into a data stream using information in the packet headers. UDP, on the other hand, is a packet-oriented protocol where the application itself divides the data into packets and sends them to the other end. The other end does not have to reassemble the data into a stream. Note, some applications might indeed present the data as a stream when the underlying protocol is UDP. However, this is the layering of an additional protocol on top of UDP, and it is not something inherent in the UDP protocol itself.
- **TCP Is a Reliable Protocol, While UDP Is Unreliable?** — The packets that are sent by TCP contain a unique sequence number. The starting sequence number is communicated to the other side at the beginning of communication. Also, the receiver acknowledges each packet, and the acknowledgment contains the sequence number so that the sender knows which packet was acknowledged. This implies that any packets lost on the way can be retransmitted (the sender would know that they did not reach their destination because it had not received an acknowledgment). Also, packets that arrive out of sequence can be reassembled in the proper order by the receiver.

Further, time-outs can be established, because the sender will know (from the first few packets) how long it takes on average for a packet to be sent and its acknowledgment received. UDP, on the other hand, simply sends the packets and does not keep track of them.

Thus, if packets arrive out of sequence, or are lost in transmission, the receiving end (or the sending end) has no way of knowing.

TCP communication can be compared to a telephone conversation where a connection is required at all times and two-way streaming data (the words spoken by each party to the conversation) are exchanged. UDP, on the other hand, can be compared to sending letters by mail (without a return address). If the other party is not found, or the letter is lost in transit, it is simply discarded. The analogy fails, however, when considering the speed of communication. Both TCP and UDP communication roughly happen at the same speed, because both use the underlying Internet Protocol (IP) layer.

### ***4.1.3 Reason for choosing UDP***

UDP was chosen as the xPC Target transport layer because of its lightweight nature. Since the primary objective of an application running in the xPC Target framework is real-time, the lightweight nature of UDP ensures that the real-time application will have a maximum chance of succeeding in real-time execution. Also, the datagram nature of UDP is ideal for sending samples of data from the application generated by the Real-Time Workshop software. Because TCP is stream oriented, separators between sets of data must be used for the data to be processed in samples. It is easier to build an application to deal with unreliable data than it is to decode all of this information in real-time. If the application is unable to process the data as quickly as it arrives, the following packets can just be ignored and only the most recent packet can be used.

Communication can involve a packet made up of any Simulink data type (double, int8, int32, uint8, etc.), or a combination of these. The xPC Target block library provides blocks for combining various signals into one packet (packing), and then transmitting it. It also provides blocks for splitting a packet (unpacking) into its component signals that can then be used in a Simulink model. The maximum size of a packet is limited to about 1450 bytes.

### ***4.1.4 Note on UDP Communication***

The UDP blocks work in the background when the real-time application is not running. The UDP communication has been set up to have a maximum of two UDP packets waiting to be read. This applies to one UDP port, which corresponds to one UDP Receive block. All subsequent packets are rejected. This prevents excessive memory usage and minimizes the

load on the TCP/IP stack. Consequently, when any large background task is performed, such as uploading a screen shot or communicating large pages through the WWW interface, packet loss might occur. Design applications so that this is not critical. In other words, the receipt of further packets after the ones that were lost ensures seamless continuation.

Note also that because UDP block transfers operate as background tasks, the xPC Target software disables them in polling mode.

## 4.2 Visualization

An important aspect of the real-time environment is visualization. With the help of a visualization screen, the assembly of a constructed model can be checked as well as the motion of the model that is being simulated. Possible errors that were made constructing the model (especially when it comes to composition of bodies) are noticed at a glance. Figure 4.1 shows the visualization of the Hydraulic Log Loader.



*Figure 4.1. Visualization of The Hydraulic Log Loader*

SimMechanics offers different ways of visualizing models. First of all, a default view can be used in which only 'centers-of-gravity', 'coordinate-frames' and 'body-geometries'

(geometries according to specified coordinate-frames) (see figure 2.1). Since the R2008b-release of Matlab, SimMechanics offers an option to attach user-made .stl-graphics to bodies. During a simulation, the model can be turned, panned and zoomed to investigate a particular component. The animation can be stored into a .avi-file.

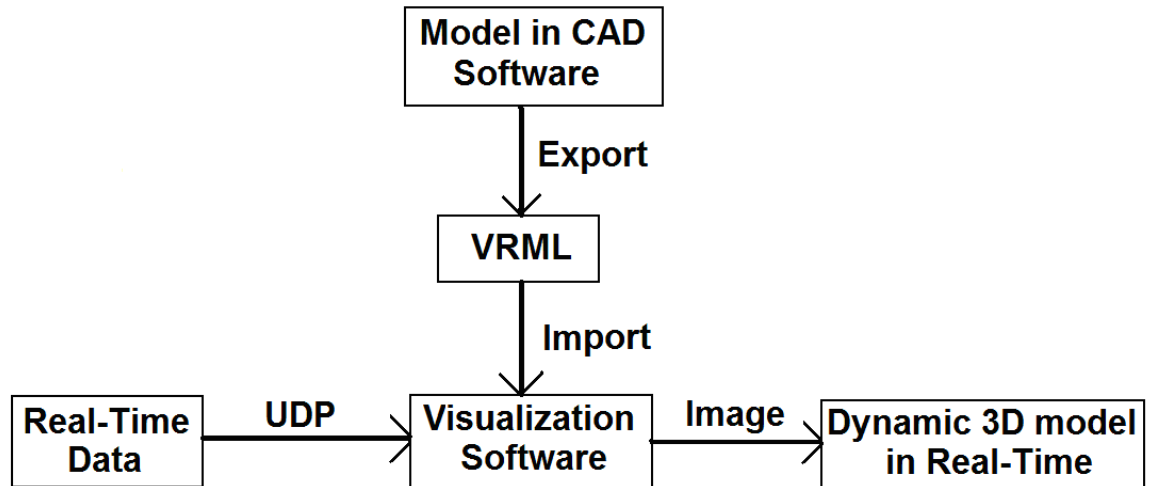
There are many methods to visualize the real-time mobile machine. For example, the VR Sink in Matlab, the above mentioned way and some visualization software edit by some research institute and university, and so on. The visualization software used in this thesis is MEC visual developed by the Department of the Mechanics and Design, Tampere University of Technology. However, no matter what kind of visualization software used, the principle behind that is same, which is based on VRML.

VRML (Virtual Reality Modeling Language) is a standard file format for representing 3-dimensional (3D) interactive vector graphics, designed particularly with the World Wide Web in mind. It is a text file format where, e.g., vertices and edges for a 3D polygon can be specified along with the surface color, UV mapped textures, shininess, transparency, and so on.[15] VRML is a specification for displaying 3-dimensional objects on the World Wide Web. You can think of it as the 3-D equivalent of HTML. Files written in VRML have a.wrl extension (short for world).

VRML files are commonly called "worlds" and have the \*.wrl extension (for example island.wrl). Many 3D modeling programs can save objects and scenes in VRML format, like Catia and Solid Works. First of all, build model in 3D software, like Catia in this project. Secondly, export VRML file from Catia. Finally, import this file into visualization software.

Figure 4.2 shows the principle behind Communication and Visualization





*Figure 4.2. The Principle of Communication and Visualization*

## **5. IMPLEMENTING THE VIRTUAL REALITY SYSTEM AND DISCUSSING THE RESULT**

This thesis focuses on the development of the real-time simulation environment, and uses the logloader as one example to illustrate how to integrate the virtual reality environment. The following sections will show it step by step.

### **5.1 Model Building in SimMechanics**

Figure 5.1 shows the SimMechanics model of the logloader which is the subsystem of the manipulator. Inside it, the blue boxes are cylinders which are the subsystems like the figure 5.2 describes. The yellow boxes are input signals like the figure 5.3 shows.

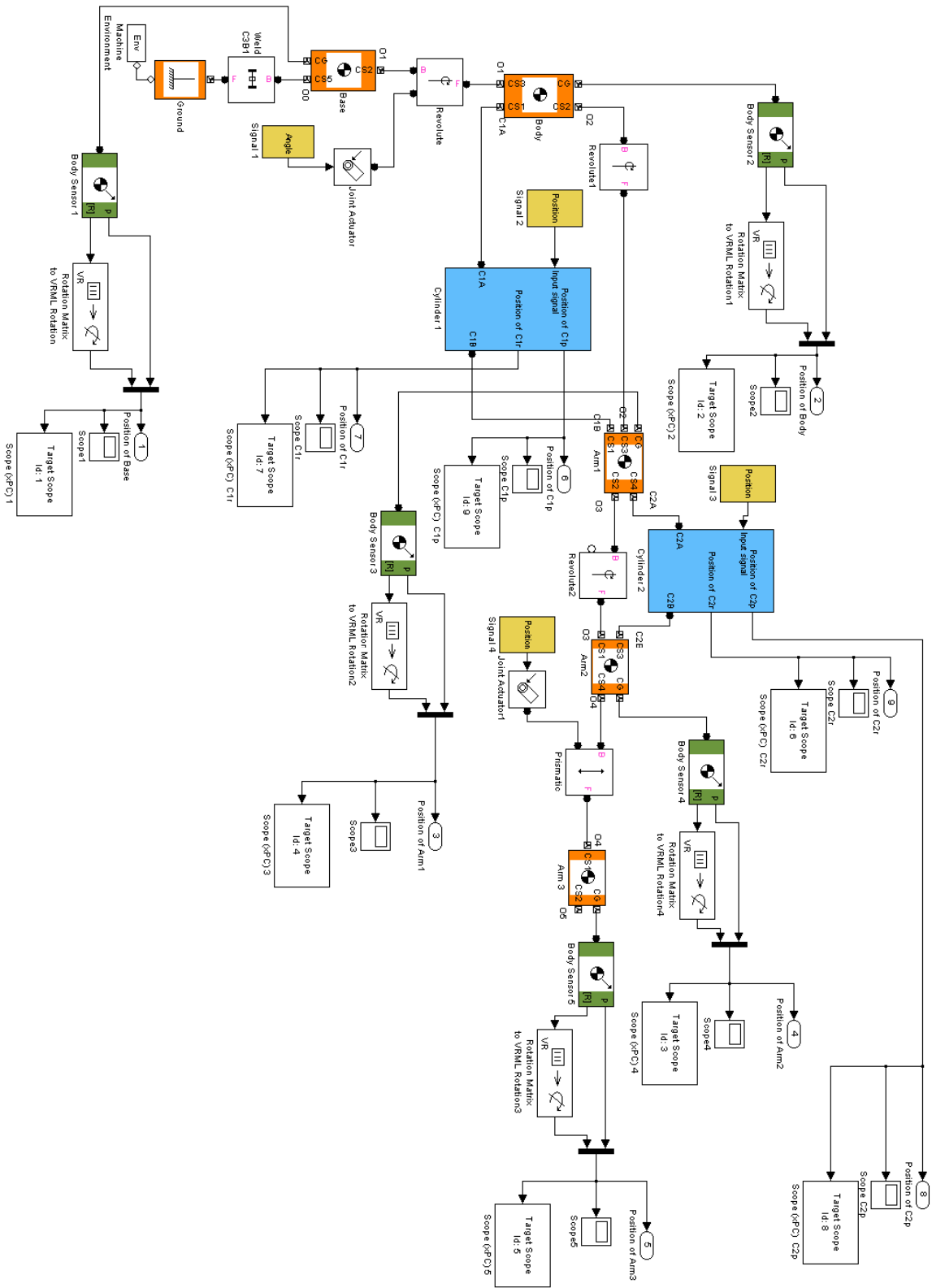


Figure 5.1. SimMechanics Model for The Logloader

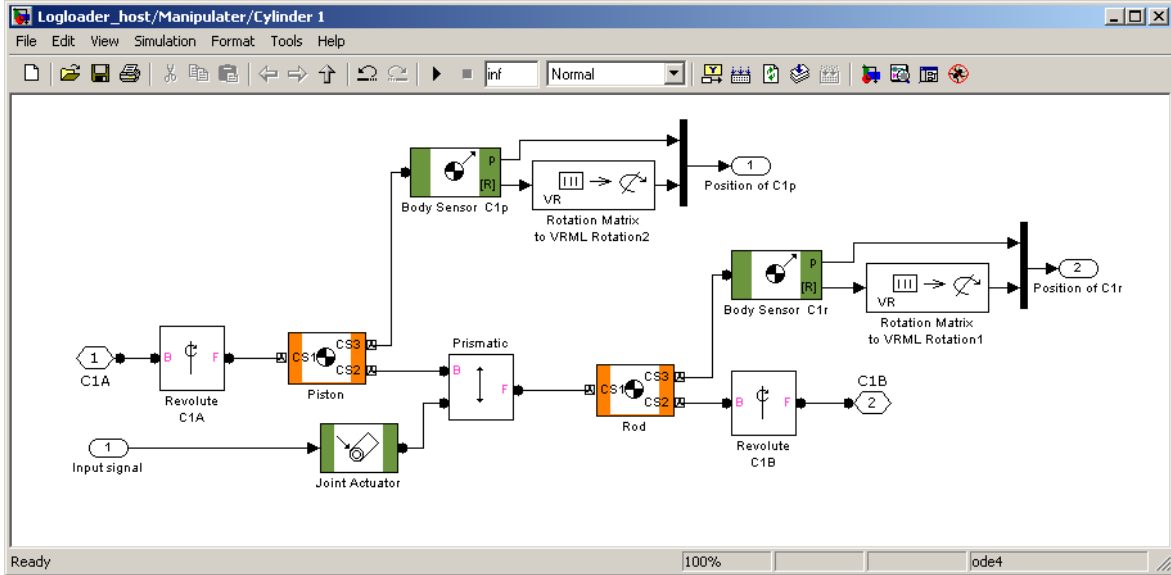


Figure 5.2. The Subsystem of Cylinder 1

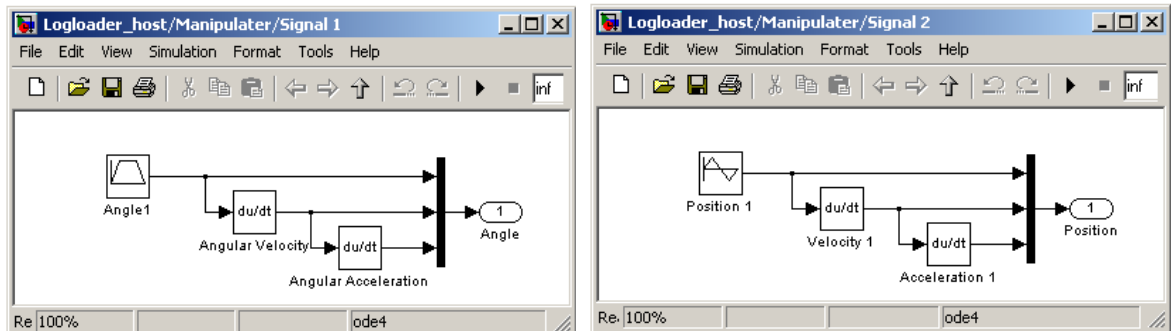


Figure 5.3. The Subsystem of Input Signals

## 5.2 Booting the target PC

xPC Target Explorer is a graphical user interface for the xPC Target product. It provides a single point of contact for almost all interactions. Through xPC Target Explorer, we can make a CD to boot the target PC and configure the host PC for the compiler.

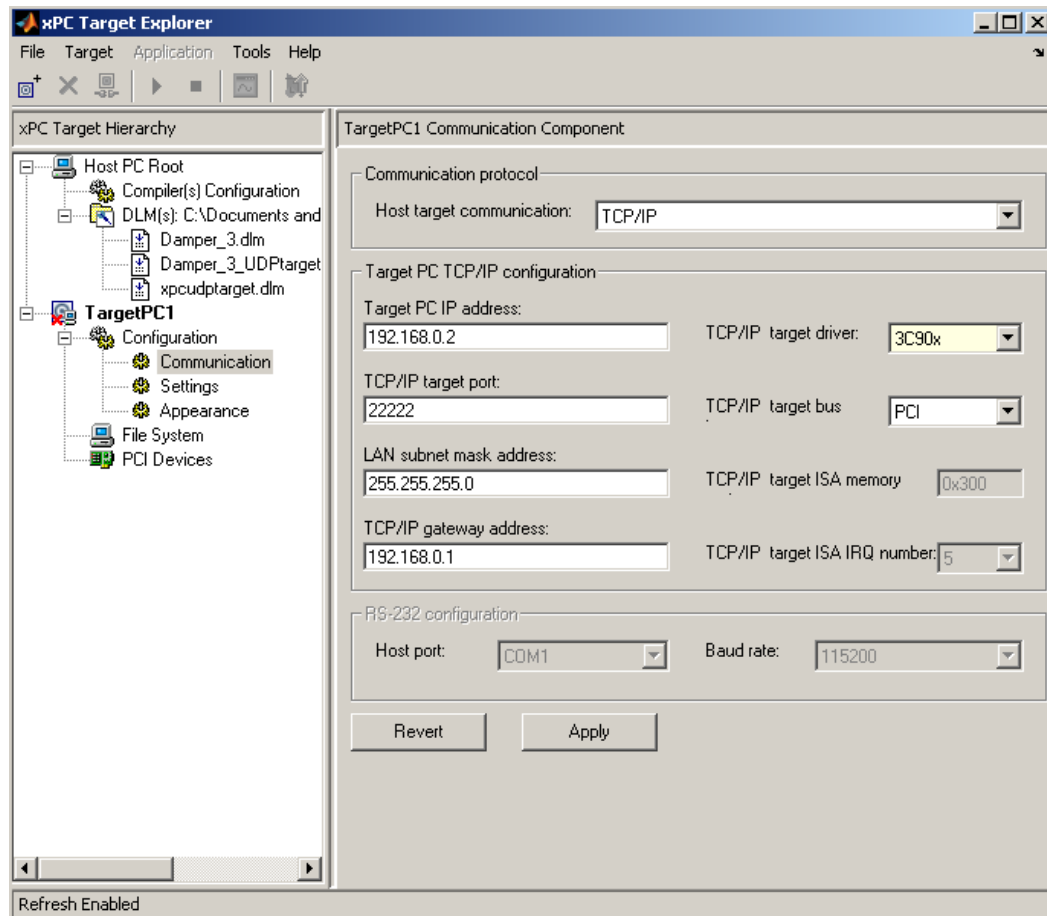


Figure 5.4. The xPC Target Explorer Window

Then burn a CD and boot the target PC.

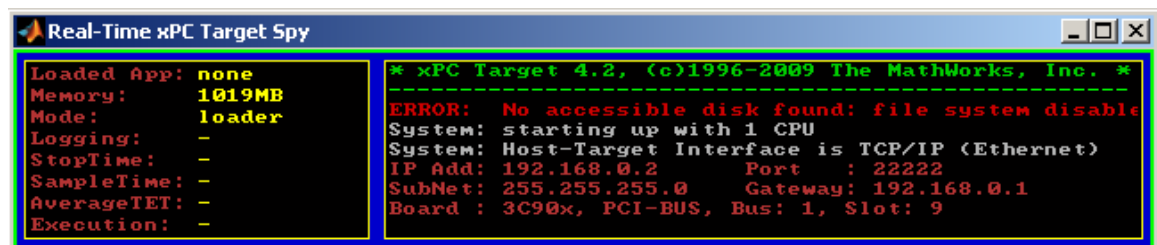


Figure 5.5. xPC Target Boots, The Kernel and Display Information on The Target PC

### 5.3 Downloading the model into the target PC

I use the xPC Target build process to generate C code, compile, link, and download my target application to the target PC. By default, the build procedure downloads the target

application to the default target PC, as designated in the xPC Target Explorer. In the Simulink window and from the **Tools** menu, select **Real-Time Workshop**. From the **Real-Time Workshop** submenu, click **Build Model**. After the compiling, linking, and downloading process, a target object is created in the MATLAB workspace.

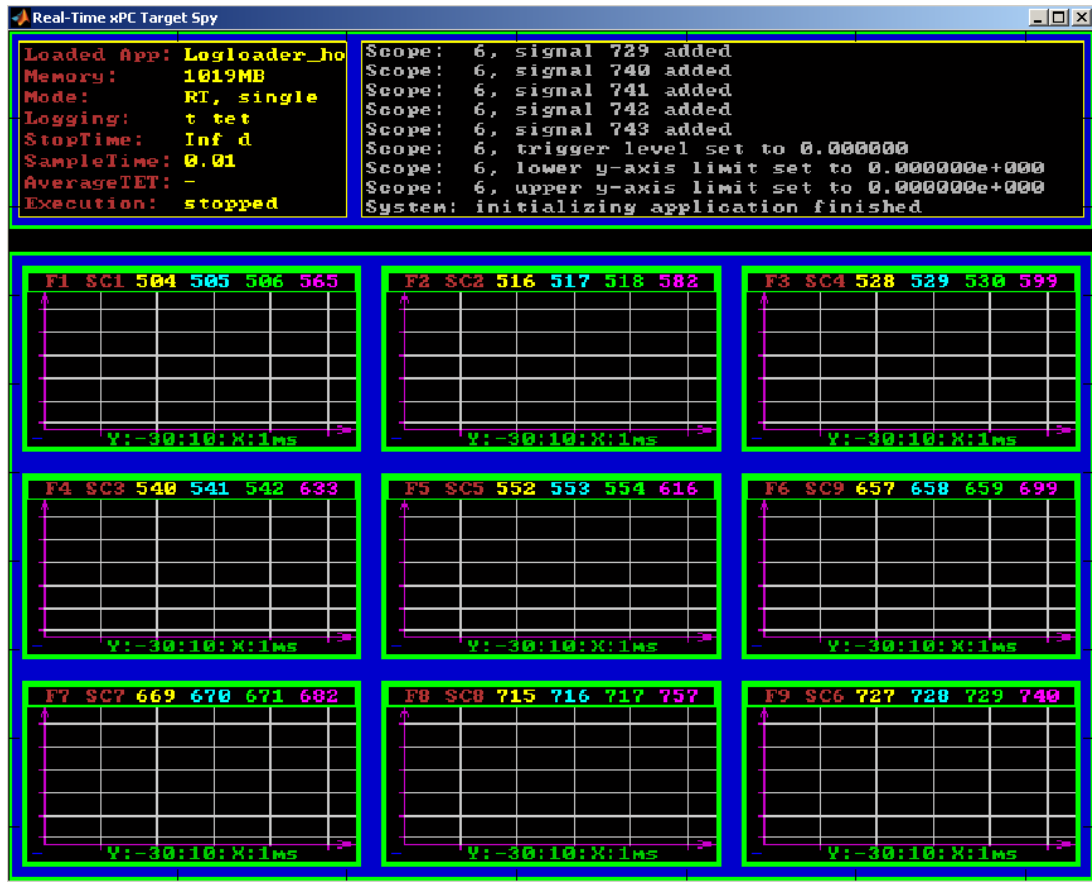


Figure 5.6. The Model in The Target PC

The next task is to run the target application in real time on the target PC.

## 5.4 Simulating in Real-Time

Although the sample time can be changed between different runs, we can only change the sample time without rebuilding the target application under certain circumstances. If the sample time chosen is too small, a CPU overload can occur.





Figure 5.8. The Model Running in The Target PC and The Plots of Motion

## 5.5 Communication

UDP was chosen as the xPC Target transport layer in this thesis. It is available in xPC Target toolbox of Simulink Library Browser.



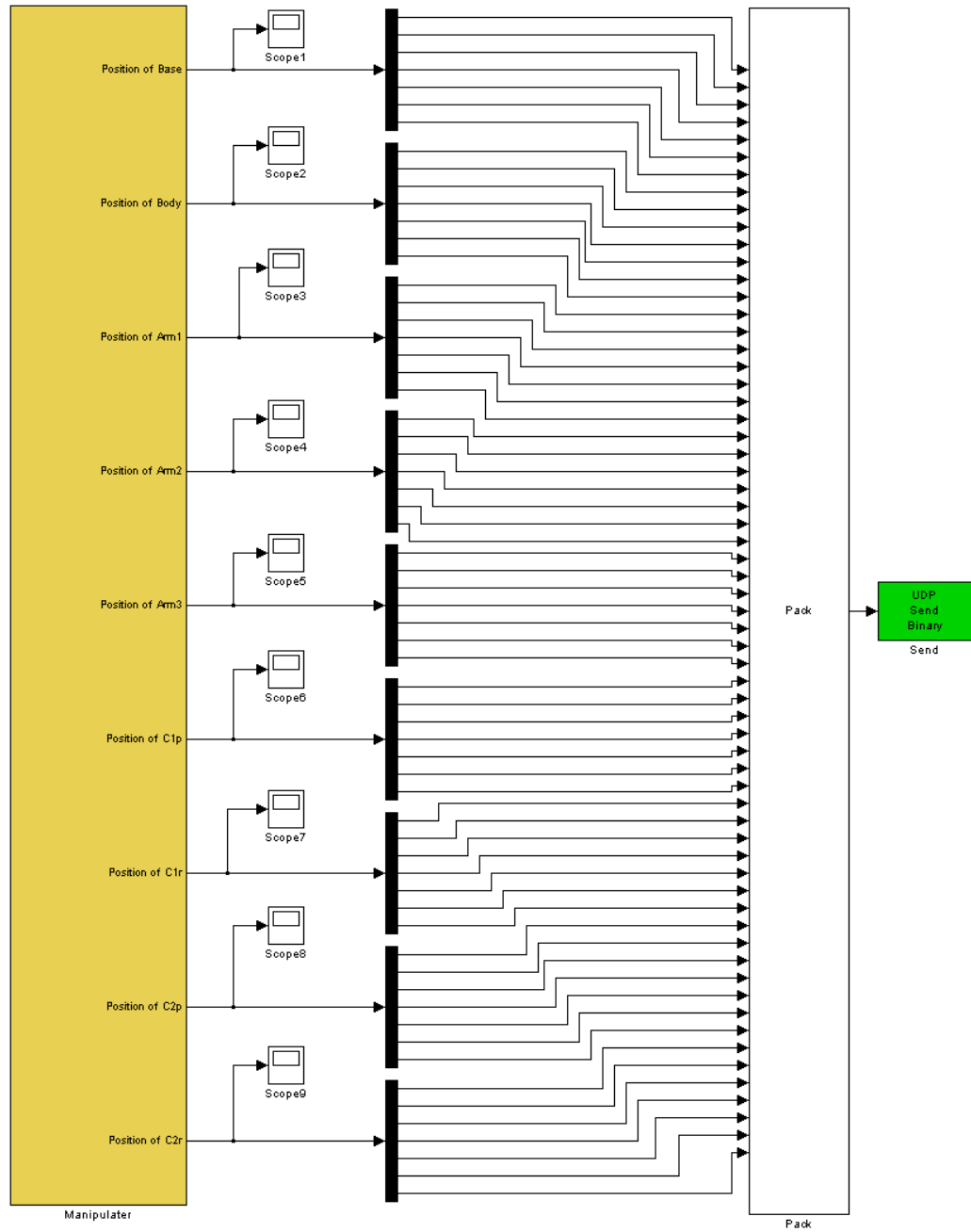


Figure 5.9. Datas Sent to The Host PC

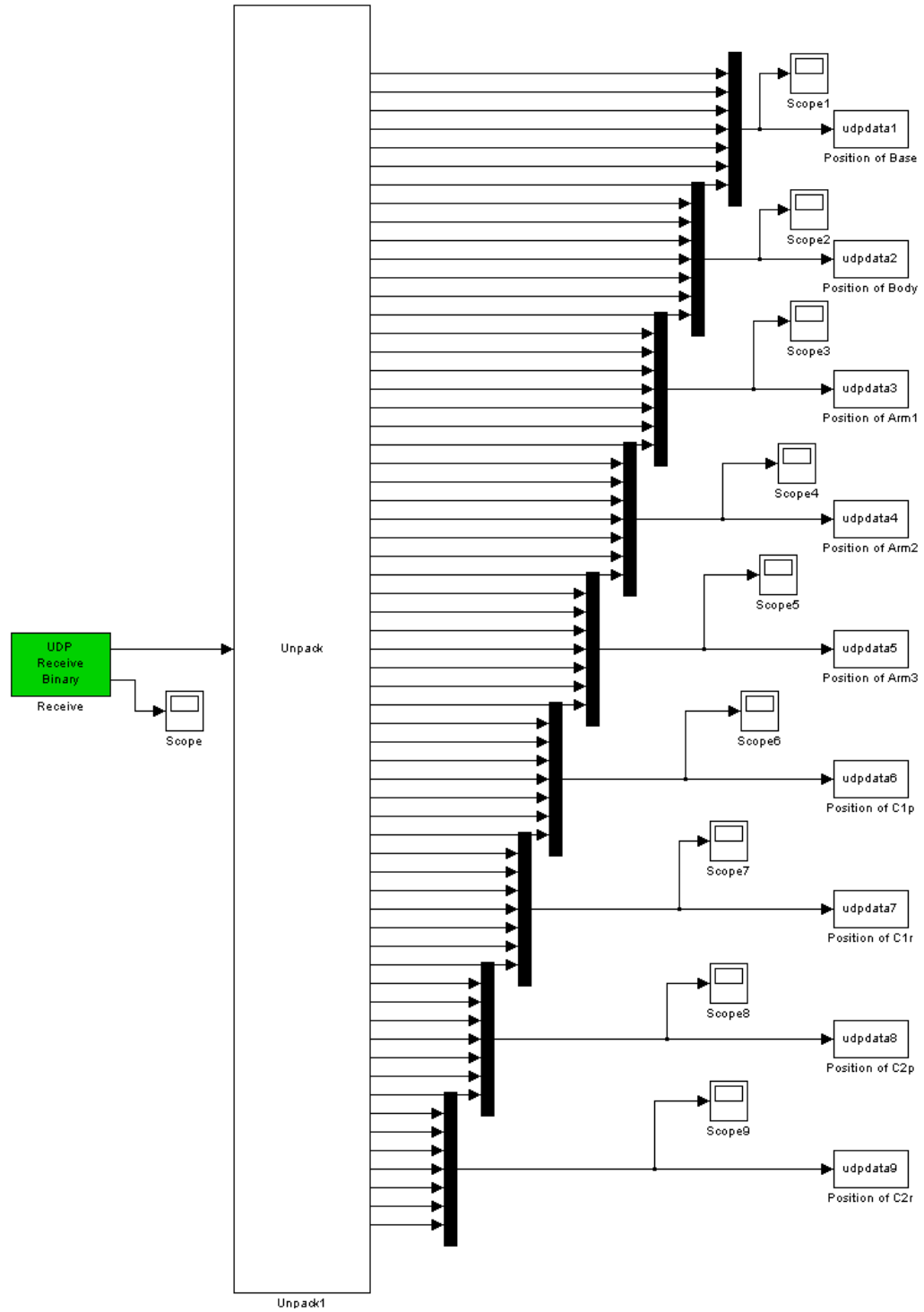
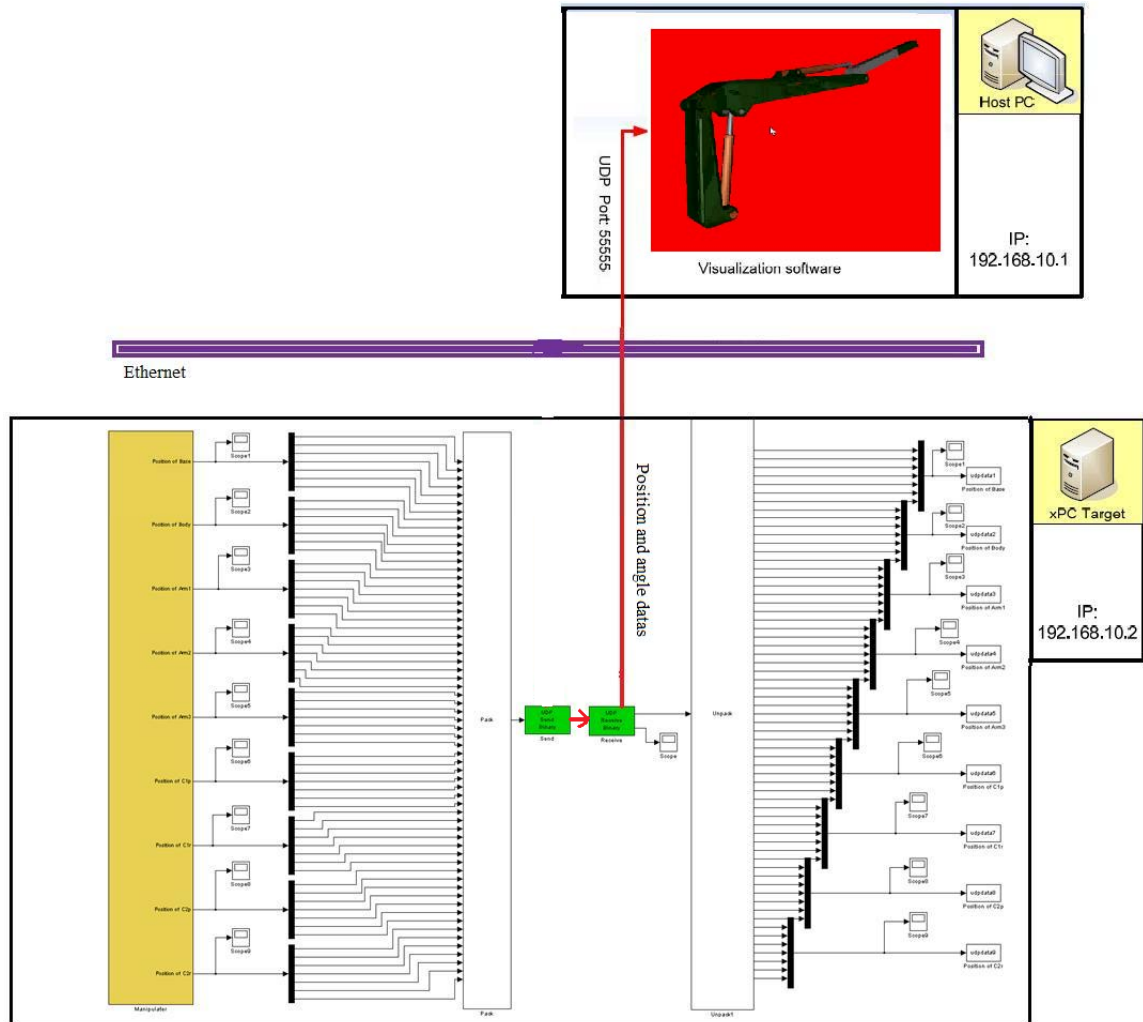


Figure 5.10. Datas Received from The Target PC

## 5.6 Visualization

Finally, through the visualization software, we could observe the dynamic motion of the mobile machine in real-time.



*Figure 5.11. The Real-Time Environment*

Figure 5.11 shows how to integrate the SimMechanics, xPC Target, UDP and visualization software into a real-time environment.

## 5.7 Discussion

As mentioned in section 2.3, some assumptions were made for the simplified model. This project task did not include how to model the system, and the model was therefore assumed

correct throughout the project. However, links between the joint  $C2_B$  and  $O_3$  were changed after the model had been verified, and this might have caused an inaccuracy in the model of the real system. Although the image exported from an existing model which is built in Catia is just for visualization, the size of the parts still need to verify. That means they should be as same as the parameters of the model in Matlab. If they are not, it may cause the movement of the visualization model will deviate the preconcerted path. Compared to simulation runs, some tuning of weighting parameters was necessary in order to implement the Mobile Machine for real-time use without hurting the constraints. This fact leads to the recommendation that a verification of the model needs to be carried out.

The Real-Time Workshop has the advantage of easy conversion from a Simulink and SimMechanics model to real-time experiments. It adds the capability of acquiring data in real time, and is supposed to immediately process them by MATLAB commands or a Simulink and SimMechanics model in order to send them back to the outside world. However, it does require a lot of CPU-power in the data acquisition and processing. The PC was driven by a 2.66 GHz Core™2 processors with vPro™ technology, and MATLAB had to be given top priority in the Windows Task Manager. By looking at the plot of the response in section 5.4, it is clearly seen that the system has a delay. This corresponds to more than one sampling time, which contributes to question the toolbox's capability of implementing in real-time.

## 6. CONCLUSIONS AND FUTURE ENHANCEMENT

All the components of the system worked in good synchronization so that the users got a “realistic” enough experience in driving the log loader despite using Windows as platform for most of the program components.

Recommendations were also given, on the possibilities of use for the environment and on what modifications need to be made to make the environment safer, more realistic and user friendly.

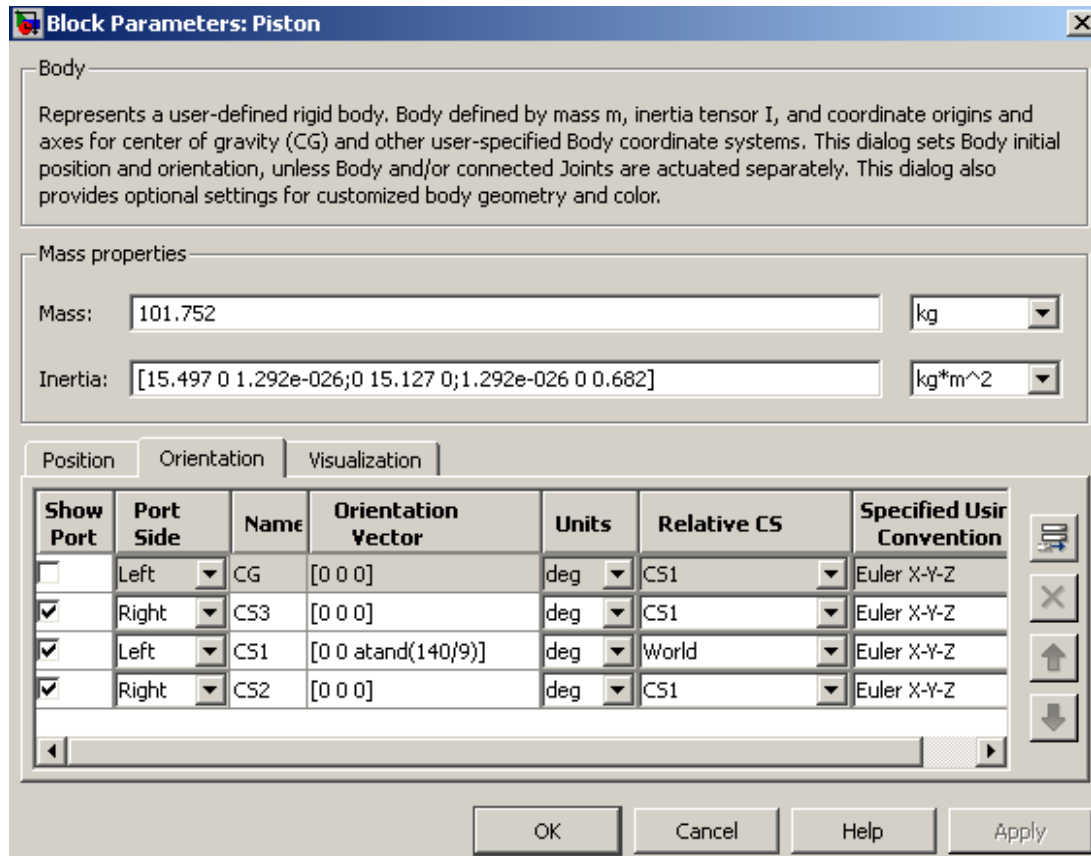
Integration and running of the Mobile Machine in Virtual Reality environment was done successfully and it was noted that the use of the xPC target in real-time simulation is possible, but more research is needed in this subject.

### 6.1 Conclusions

This thesis has investigated the possibility of building a real-time control model for a mobile machine using Matlab. The primary objective of this project was to extend the Virtual Reality environment for real-time use. The basic Mobile Machine algorithm was coded in “C” and interfaced with Simulink by use of a S-function structure. Two tools for implementation of Mobile Machine for real-time use was tested, namely the SimMechanics and xPC Target. In addition, the toolbox caused a system overload when the execution of the algorithm required more than one sampling period. The implementation of Mobile Machine by this alternative is demanding on excessive amounts of CPU-power. In order to meet these demands, a PC with a 2.66 GHz processor was used. However, even though Matlab was given top priority in the system manager, the complexity of calculations had to be limited. The problems with system overload leads to the conclusion that the toolbox is not able to implement a realistic Mobile Machine in real-time exactly at present PC technology.

In Chapter 2, the model built in SimMechanics. Each parameter should be set carefully,

because the correctness of the model will affect the result and the visualization of the whole system directly. For example, if the parameters of the piston of the cylinder, especially the orientation parameters, are not set correctly, it may cause the disorder of the position of the cylinder.



**Figure 6.1.** The Parameters of The Piston

In Chapter 3, the use of the xPC Target from MATLAB to support this real-time application has been introduced in the beginning of the chapter. Booting the target PC should be treated carefully. The type of kernel in target PC must be checked first, then it could be booted successfully.

In Chapter 4, the main topic of it is the design of a communication module for the Mobile Machine which works on the xPC Target. The design process starts with analyzing the communication protocol of the Mobile Machine and how to connect the xPC Target to the model built in SimMechanics. Finally an interface of the system communication block was created. This interface will include the inputs and outputs of the communication block

which are UDP Send Binary and UDP Receive Binary blocks. The main problem is how to improve communication speed of xPC Target with external data exchange. Due to the computer and network technology at moment, this speed is limited. I believe that in the future it will be improved.

In Chapter 5, a real-time Mobile Machine application was designed and implemented. This application allow the Mobile Machine to move following a certain path. This path can be a constant or a mathematic function.

During the whole work of this thesis new problems constantly arose and solutions for these problems were also discovered. Here the problems and solutions for this application will not be mentioned again. In the next section some suggestions for the future work will be given.

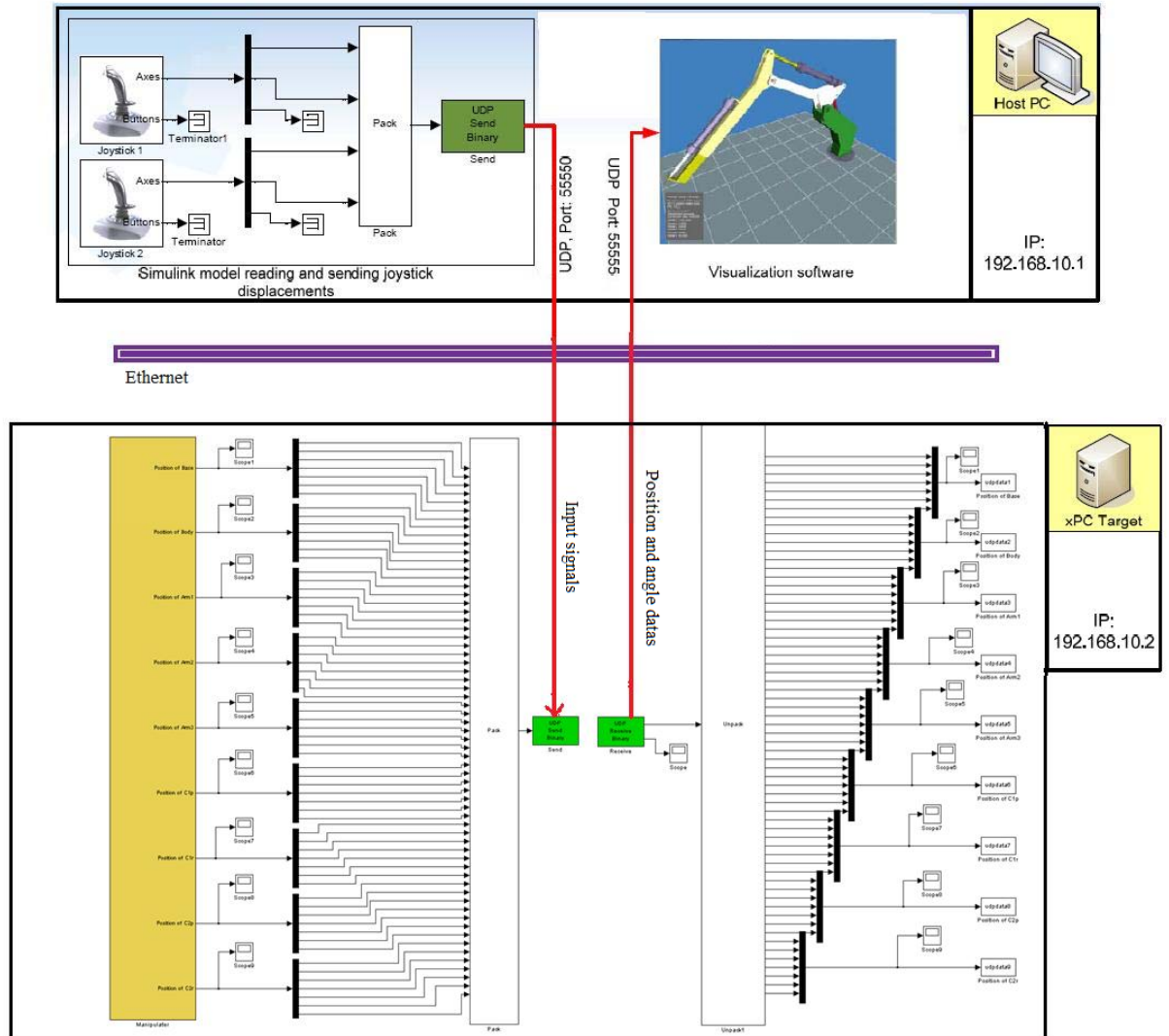
## **6.2 Suggestions for the Future Work**

There are a lot of enhancements that can be used to improve or enhance this real-time environment.

Target of the project was the integration of a real-time environment. Project was a success as the real-time environment was built successfully and featuring modularity that enables the environment to be modified to whatever need desired.

Below are suggestions on how to make the current version of the real-time environment more user-friendly, safer and add the level of realism for other parts then the motion platform.

- 1) Higher quality joysticks with full force feedback for input signals.



**Figure 6.2.** *The Whole Model Expanded by Joysticks*

2) Addition of a VR glove, possibly with haptic feedback, a 3D display and the CAVE which is a surround-screen, surround-sound, projection-based VR system to the laboratory equipment as these would give better immersion level if simulator would be used for prototype testing of user interfaces.

Other suggestions on the numerous possible uses of the real-time environment:

- Development of training systems
- Testing of 3D models and dynamic solvers.
- Design prototyping



- User interface testing and development
- Recording and research of user habits and actions when driving a vehicle or using a device/machine.

## REFERENCES

- [1] Burdea, Grigore; Coiffet, Philippe. Virtual Reality Technology, Wiley, New Jersey, Second Edition, 2003
- [2] Jani Heikkinen, Integration of Real-Time Simulator, Motion Platform and Haptics, Lappeenranta University of Technology, 2009
- [3] Loglift Oy Ab, Loglift 140s, [www.loglift.com](http://www.loglift.com)
- [4] E. J. Haug, Computer Aided Kinematics and Dynamics of Mechanical Systems, Volume 1, Basic Methods, Allyn and Bacon, 1989
- [5] T. R. Kane, D. A. Levinson, Multibody Dynamics, Journal of Applied Mechanics, 50, 1983, pp. 1071-1078
- [6] R. E. Roberson, R. Schwertassek, Dynamics of Multibody Systems, Springer-Verlag, 1988
- [7] W. O. Schiehlen, Multibody System Handbook, Springer-Verlag, 1990
- [8] A. A. Shabana, Dynamics of Multibody Systems, Wiley, 1989
- [9] Lawrence F. Shampine, Mark W. Reichelt, Jacek A. Kierzenka, Solving index-1 DAEs in MATLAB and Simulink, SIAM Review, 41, 1999, pp. 538-552
- [10] Michael Schlotter, Multibody System Simulation with SimMechanics, Darmstadt University of Technology, 2003, pp.14-15
- [11] [http://www.physnet.uni-hamburg.de/physnet/matlab/help/toolbox/physmod/mech/mech\\_building2.html#119263](http://www.physnet.uni-hamburg.de/physnet/matlab/help/toolbox/physmod/mech/mech_building2.html#119263)

- [12] Kai Wu, B.Eng, Real-time Control of a Mobile Robot Using Matlab, The University of Applied Science Hamburg , 2004, pp.7, pp.9, pp11
- [13] The MathWorks: Real-Time Workshop Getting Started
- [14] The MathWorks: xPC Target For Use with Real-Time Workshop User's Guide(Version 2)
- [15] "Version1.0 Specification". Web3d.org.  
<http://www.web3d.org/x3d/specifications/vrml/VRML1.0/index.html>. Retrieved 2010-02-23.

# APPENDIX

## List of Figures

1.1	A Driving Simulator (Carlton)	2
1.2	Host PC and Target PC in This Project	3
1.3	The Hydraulic Log Loader Loglift 140s	5
1.4	Datas Transfer from The Target PC to The Host PC	6
2.1	A MATLAB Visualization of The Hydraulic Log Loader	9
2.2	The Simulink Models	11
2.3	The SimMechanics Models	11
2.4	The Abstract Graph of The Hydraulic Log Loader Loglift 140s	14
2.5	The Interface between SimMechanics Blocks and Simulink Blocks	18
2.6	The Block Parameters of The Crane	18
2.7	The Subsystem of The Hydraulic Log Loader Model	19
3.1	Real-Time Workshop Code Generation Process	24
3.2	Hardware Connection between Target PC and Host PC	26
3.3	Configuration of Target Boot Disk	28
3.4	xPC Target Boots, The Kernel and Display Information on the Target PC	28
3.5	Loading a Matlab SimMechanics Model to The Target PC	29
4.1	Visualization of The Hydraulic Log Load	33
4.2	The Principle of Communication and Visualization	35
5.1	SimMechanics Model for The Logloader	37
5.2	The Subsystem of Cylinder 1	38
5.3	The Subsystem of Input Signals	38
5.4	The xPC Target Explorer Window	39
5.5	xPC Target Boot, the Kernel and Display Information on The Target PC	39
5.6	The Model in The Target PC	40
5.7	The Control Window	41

5.8	The Model Running in The Target PC and The Plots of Motion	42
5.9	Datas Sent to The Host PC	43
5.10	Datas Received from The Target PC	44
5.11	The Real-Time Environment	45
6.1	The Parameters of The Piston	48
6.2	The Whole Model Expanded by Joysticks	50