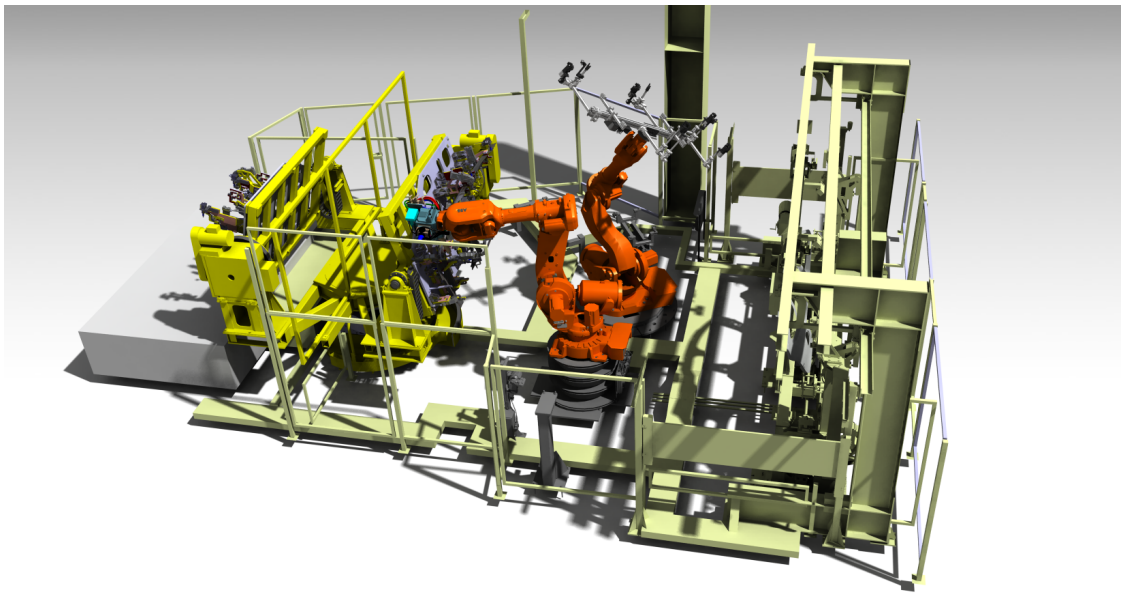
CHALMERS



Virtual Commissioning of an Existing Manufacturing Cell at Volvo Car Corporation Using DELMIA V6

Master's Thesis in the Master Degree Programme
Systems, Control and Mechatronics

Heidari Ali
Salamon Oliver

Automation Group
Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2012
Report No. EX023/2012

MASTER'S THESIS

Virtual Commissioning of an Existing Manufacturing Cell at Volvo Car Corporation Using DELMIA V6

Master's Thesis under Master Degree Programme

Ali Heidari - Oliver Salamon

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2012

Abstract

Virtual commissioning is a process which allows a comprehensive evaluation of production systems before performing physical commissioning. The programmable logic controller (PLC) code can be debugged before using it in a real production system. A growing number of companies have recently started taking interest in this technology as it reduces the time and cost of introducing new products and different scenarios can be performed to validate the manufacturing controllers in the virtual environments prior to the physical commissioning.

This work discusses an approach to create and simulate an existing Body-in-White manufacturing workcell in Volvo Car Corporation (VCC) using Dassault Systèmes digital manufacturing and production solution DELMIA V6. Also the migration process of other 3D model formats such as Process Simulate, CATIA V4, and V5 into DELMIA V6 has been investigated. It is shown that it is possible to create a simulation using the given resources; all models could be used. Also, a large amount of the robot functionality could be imported after editing certain functions which are not supported. One of the substantial properties for a successful simulation model is to replicate the actual controller, robot programs and HMI applications. These aspects of the Dassault Systèmes PLM solution are validated in this project. The latest release of DELMIA V6 is also tested in terms of connectivity to an actual PLC through the OPC communication protocol.

Keywords: Virtual Commissioning, Product Lifecycle Management, DELMIA V6, 3D-modeling, PLC, RAPID programs, Volvo Car Corporation cell.

Contents

Acknowledgements

We would like to acknowledge and extend our heartfelt gratitude to the following persons, because without their effort, the completion of the project would not have been possible.

Foremost we would like to thank our academical supervisor at Chalmers, Mohammad Reza Shoaei, for honestly taking interest in our thesis, always being available to help us and discuss different problems, offer constant supervision and for always making sure everything is well with the project. We would also like to thank our academical project examiner, Petter Falkman for entrusting us with this thesis, for your guidance and constructive comments, and for helping with all necessary arrangements.

We are also very grateful to our industrial examiner, Stefan Axelsson at Volvo Car Corporation in Torslanda for keeping in touch with us during the whole project, providing the Mitsubishi PLC and sharing his expertise by providing a lot of help regarding the models, robots and the PLC. We also appreciate that he found the time to invite us over to the Torslanda factory several times and gave us a very interesting factory tour. We are also very thankful to Håkan Pettersson for his help, cooperation and providing the OPC program for our project.

From the Dassault Systèmes side, we would like to express our sincere gratitude to our industrial supervisor Anthony Hairon for the very comprehensive support with the DELMIA side of the project and for spending many hours and late evenings helping us whenever it was necessary. Also our thanks to Hans Eriksson at Dassault Systèmes in Göteborg and Edouard Pardieu for their contribution which made this project possible.

Göteborg, 2012
Ali Heidari
Oliver Salamon

1 Introduction

Manufacturing systems can be analyzed and verified using a digital prototype of a physical model. Production simulation is performed in different ways depending on the problem type which is going to be studied. There are different names for these methods depending on the business or organization. Some of these names are: Virtual Manufacturing, Virtual Commissioning, Digital Production and Digital Plant.

Product Lifecycle Management (PLM) is an integrated and information driven approach to all aspects of a product from its design inception through its manufacture, deployment and maintenance, culminating in its removal from service and final disposal [15]. PLM integrates people, data, processes and business systems and provides a product information backbone for companies and their extended enterprise [11][6].

Depending on the industries size and field of activities different solutions can be used to implement the PLM in different ways. For instance, Teamster AB in Gothenburg used the free program Google Sketchup to visualize their projects in the Teamster Robotic Automation department. Google SketchUp is a 3D modeling program which is designed for architectural, civil, and mechanical engineers as well as film makers, game developers, and related professions [4], [2]. However Teamster also uses traditional CAD tools which they are missing in Google SketchUp.

Targeting the mid-size PLM solution, SAP which stands for "Systems, Applications, and Products in Data Processing", was founded in 1972. The SAP Product Lifecycle Management (SAP PLM) application provides full support for all product-related processes - from the first product idea, through manufacturing to product service [8]. SAP has a few disadvantages when it comes to the PLM market. It has some difficulties with the BOM (Bill of Material) of life cycle management. SAP's messaging is also unclear and does not take into account the PLM functionality that is distributed throughout many applications [3].

Most of the major manufacturing companies in the automotive and trucking industry, such as Volvo Trucks, Volvo Car, Scania and BMW, are using enterprise solutions such as Dassault Systèmes DELMIA [1] and Siemens Teamcenter Process Simulate [7]. These software give the user more opportunities to validate the feasibility of an assembly process by checking reachability and collision clearance. These software allow manufacturers to virtually define, plan, create, monitor and control all production processes.

Today, the development of PLM software has encouraged companies to virtually build and simulate their production processes. This makes it possible to have the physical attributes of the production right from the start. Robots and their program can be simulated and verified in a virtual environment, for instance, Dassault Systèmes DELMIA has the ability to import and modify the robot programs from various robot manufacturers. Actual PLC programs can be evaluated and tested

against different scenarios using the virtual manufacturing model connecting via the OPC protocol.

Often writing the PLC program comes as the last phase in the construction of a production cell, when the mechanical and electrical parts are already finished. This yields a tedious and time consuming task for automation engineers to write and verify the PLC programs. Using the simulation software can help to concurrently develop and verify the PLC code while the cell is being constructed.

Virtual commissioning is the use of an accurate and realistic 3D simulation to validate the functions of production equipment control systems prior to actual implementation. The automation engineer can easily connect a Programmable Logic Controller (PLC) to the 3D model and debug the PLC code before it is placed into production [10]. Virtual commissioning environments offer engineers new opportunities for the design of complex intelligent behaviors and for the enhancement of the performance of adaptive manufacturing systems. Virtual commissioning tools are used to virtually explore new solution spaces for improving performance of mechatronic systems [13]. The need of implementing virtual commissioning is arising from the complexity of the production systems. Manufacturers and big companies prefer to examine and evaluate a system in virtual reality by virtual prototypes as much as possible before building it in the real world. In this manner they are able to find the problems and modify the production system.

Virtual commissioning solutions bring some benefits; the most important ones are the following:

- Validating that equipment meets the required cycle time
- Enabling collaboration between engineering disciplines
- Early operator training before startup
- Rapid development of standards
- Evaluation of the PLC code and the robot programs [10]

The industrial requirements for a successful PLM solution are as follows:

- **Real control code:** The actual control code for PLCs and robots as well as the HMI applications shall remain unchanged while embedding them into the virtual commissioning environment.
- **Real engineering tools:** The existing engineering tools and languages for PLC, robot and HMI applications must be kept.
- **Embeddability:** The virtual commissioning environment shall be seamlessly embeddable into existing engineering workflows.

– **Extensibility:** Due to a variety of PLCs brands available on the automation market it must be possible to support several PLC systems. This includes the extensibility to further PLCs.

– **Virtual controllers:** The developed robot and PLC programs shall run on virtual controllers which behave identical to the actual controllers [12].

In summary, PLM is a very high risk solution and implementation and companies which are evaluating it must be extra careful to select the right product from PLM markets. In this project DELMIA has been chosen to be studied. It offers solutions for control engineering and automation lifecycle management that enables validation of the PLC against a virtual machine, a cell, or an entire line and the generation of a performance analysis of these systems. In addition to testing and debugging the PLC code, DELMIA V6 allows control departments to work in parallel and share information with mechanical and electrical departments earlier in the development process allowing optimization of engineering processes.

Figure 1.1 illustrates the general structure of the Dassault Systèmes digital manufacturing solution which consists of different tools for all aspects of a production cycle. PPR, which stands for Product, Process and Resources, is Dassault Systèmes' integrated model that interlinks representations of the product, the manufacturing resources (tooling, factory, operators, etc.) and the production processes. PPR is a fundamental building block of Dassault Systèmes' PLM strategy and is found in CATIA, DELMIA, and ENOVIA. This integration means that when a user makes a product design change in CATIA, the change is reflected in the associated manufacturing processes definition in DELMIA and the resources data stored in ENOVIA. In this project only the last two element, Control Engineering and Physical Commissioning, and their connection with an actual PLC are evaluated.

The general goal of the project is to carry out virtual commissioning using existing resources such as models and programs. PLC programs, robot programs and models are to be reused whenever possible. This means evaluating how well the migration from the currently used software at Volvo Car Corporation into DELMIA V6 works. It is of interest to see how well the software handles the importing and exporting of different formats, in order to save as much work and time as possible. Most of the cell 3D models are provided in Process Simulate file format, and some individual devices are available as CATIA V4 as well as V5 models. Chapter 2 covers the methodology for creating the 3D model of the cell using DELMIA V6. Chapter 3 discusses how the robot programs were imported. In chapter 4 it is covered how the PLC program was prepared for simulation. Finally, conclusions and future work are presented in chapter 6.

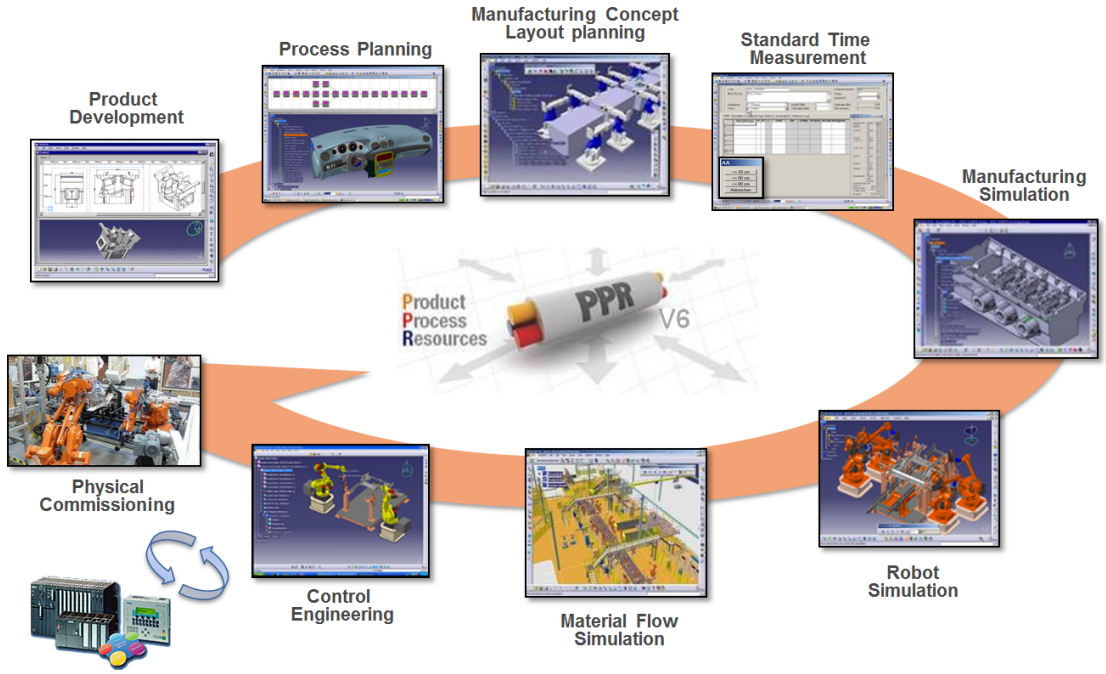


Figure 1.1: *The different phases of PLM from a DELMIA perspective*

1.1 Limitations

The actual production cell handles three different car models in the same production line; V70, S80 and V60. Only the latter car model is used for this project as it is most recent and therefore has 3D models of the production tools available in CATIA V5 formats. The pallets transporting the car sides through the factory are considered stationary in this project, hence when the welded parts are placed on them, they should be disposed of before the next parts arrive. Also four clamps in the station are closed manually by the operator, however all will be closed by the PLC in the simulation. The DELMIA versions used for the project were V6R2012 and later upgraded to V6R2012X.

1.2 Cell Description

The cell used for the project is a part of the production line at Volvo Car Corporation Body-in-White (BIW) factory in Torslanda, Gothenburg. Figure 1.2 shows a plot plan of the station and its components: Turn table, two spider assemblies, welder robot, gripper robot and transport lines. The purpose of the station is to reinforce the car sides by adding four additional parts, see Figure 1.3. It is manned by one worker that places the parts into one of two clamping platforms which are attached to a turntable. There are two different clamping platforms, one for the V60 model

and another one for both S80 and V70. There are 11 air pressure controlled clamps on the fixture, which is also referred to as spider assembly. The spider assembly has 5 retractable guiding pins, which help the operator to correctly place the parts. Two different types of sensors detect whether the parts are in place, 4 photocell sensors and 7 proximity sensors per spider assembly. The operator closes four of the clamps manually and pushes a button once all parts have been loaded. Then the automatic clamps are closed and the turntable is rotated 180°, which allows a robot with a weld gun tool to weld the pieces together through spot welding. Another robot with one of two different gripper tools is standing by until the weld operation is finished. Once ready it grabs the part and places it on a pallet at a rail which transports the car side to the next station. The cell only handles the passenger side of the car.

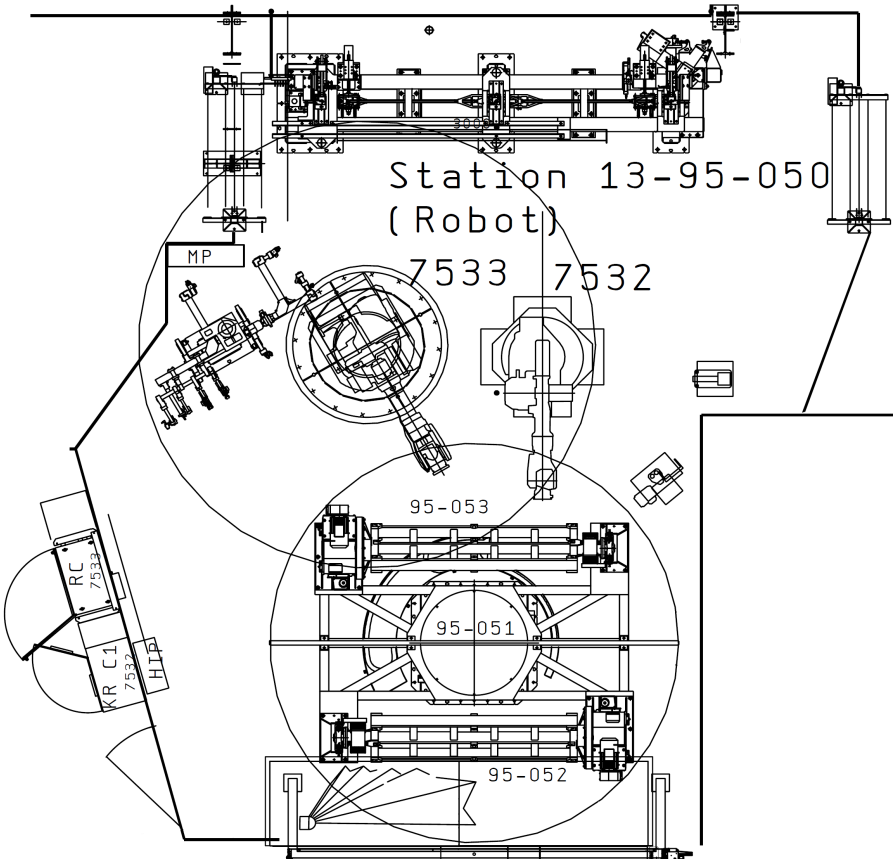


Figure 1.2: *Plot plan of the station 13-95-050*

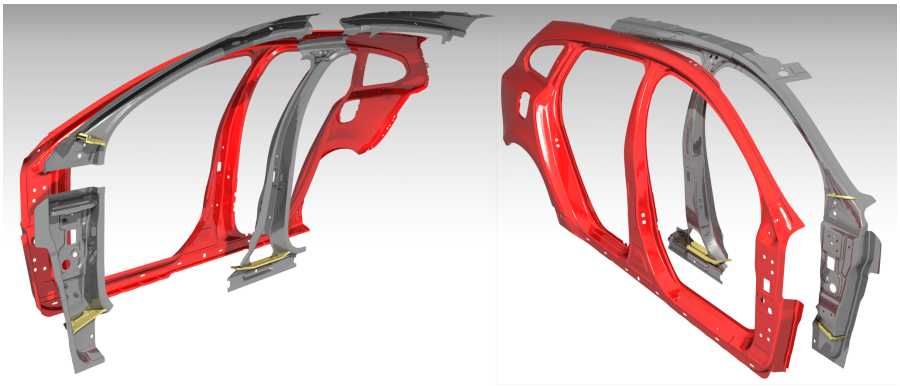


Figure 1.3: *Rendered image showing two different views of the car side (red), and the four work pieces (grey) which are welded together and placed onto the car side.*

2 Workcell 3D Model

The initial step required for creating a simulation is the creation of a 3D model. As previously mentioned one of the main goals is to utilize existing models for this project. It is possible to import a number of different formats into DELMIA. The software is also backwards compatible meaning it supports formats from previous versions. Two different types of properties of the file formats should be distinguished. Formats which are native for CATIA/DELMIA such as .model, .CATProduct and .3dxml are organized in a tree structure; they can be edited and can also consist of separate parts. The other type is the graphical representation formats such as .cgr (CATIA Graphical Representation), .stp/.step and .jt, which are a lighter non-editable version, and its geometry content is considered as one unit. This makes the format suitable for objects which are static in the station. CATIA V5 models were available for the four work pieces, the robot gripper tool used for V60 and the clamping platforms with all pins and clamps. The turntable was received as a CATIA V4 model. All the surrounding static resources such as tool stands, fences and boxes as well as the V60 car side can be extracted from an existing Process Simulate model and exported in the format .stp which is possible to import to DELMIA. No CATIA model is available for the weld gun, it was only received as a 2D drawing. However it is modeled in Process Simulate which makes it possible to export the mobile and fix components separately.

2.1 Cell Layout

DELMIA has a special workbench for simplified placing of 3D models called "Resource definition and layout". The manner in which DELMIA handles positions of models differs from that of Process Simulate. Process Simulate saves models in the native formats .psz and .zip. The .psz file stores coordinate information associated with the model and the .zip file contains the geometries. DELMIA models contain both types of information in the same file, .3dxml (CATIA V6). When using .stp models exported from Process Simulate the positioning information is lost. It is necessary to relocate the object to the correct position with available tools in DELMIA after importing. As a reference for positioning, 2D drawings can be utilized if available. It is possible to open .dwg drawing files with DELMIA V6, and measure the distances between the objects which require positioning. Another approach is to use the complete Process Simulate model as a reference. A lightweight version of the cell with only necessary geometries such as bases can be exported as a .stp file. In DELMIA this model can be used as a template to move the desired models by using the "Snap Resource" function, aligning them with the template. This approach will guarantee correct distances between objects in the cell, assuming the Process Simulate model is correct. To achieve correct "world coordinates", an origin in the model is defined. The robot programs are

based on absolute coordinates, consequently it is essential to place the origin in the correct position. Volvo uses two types of coordinates, station coordinates and factory coordinates. For this project only station coordinates were of importance, and from the CATIA models it was made clear that the origin is set at the center of the turntable, on the floor of the factory.

2.2 Creating Kinematics & Device Building

The workbench called "Device building" is dedicated to modeling mechanical systems used in manufacturing processes. To create a "smart device", joints which move have to be defined and kinematics are created. All models and sub-models such as clamps were divided into a fix part, and a mobile part. Each model consists of a varying number of components. The components which are not part of the mobile piece can be temporarily hidden, after which the remainder (fix part) can be exported as .cgr, and vice versa for the mobile parts. All these elements are then merged into one solid part. CGR models are not of as high quality as their exact view counterpart, but they consume much less memory during the generation [9]. This is useful when dealing with assemblies involving large amounts of data, and also simplifies when defining movements. The fix and mobile parts can be added together into one product file to follow the advocated tree structure of CATIA. Different types of joint movements created with the tool called "Engineering Connection". An engineering connection is based on a set of constraints between the involved components such as Products, 3D parts or shapes [9]. The most straightforward approach to create the desired kinematics is by using predefined axis systems. A tool in CATIA V5 called "Frames of interest" allows creating axis systems in .cgr models, on exact locations such as the midpoint or the corner of a face. One axis system is required on the mobile part and another on the fix part. With the Engineering Connection tool it is possible to choose which type of joint movement is desired, such as prismatic or revolution. The engineering connection is created by selecting the axis for the movement, and choosing which of the elements should be the moving one, as well as the range of the movement. An example of this can be seen in Figure 2.1.

To enable control of the device a "Motion controller" is added. This allows defining desired states of the mechanism, called "home positions". Interesting home positions are the start position and the final position, for instance when a clamp is closed and opened. A "task" defines the motion of going from the current position to a chosen home position. The motion controller also allows the control of several devices at once, as is the case with the clamps and centering pins on the spider assembly. A motion group defines which devices are to be controlled together and have common tasks.

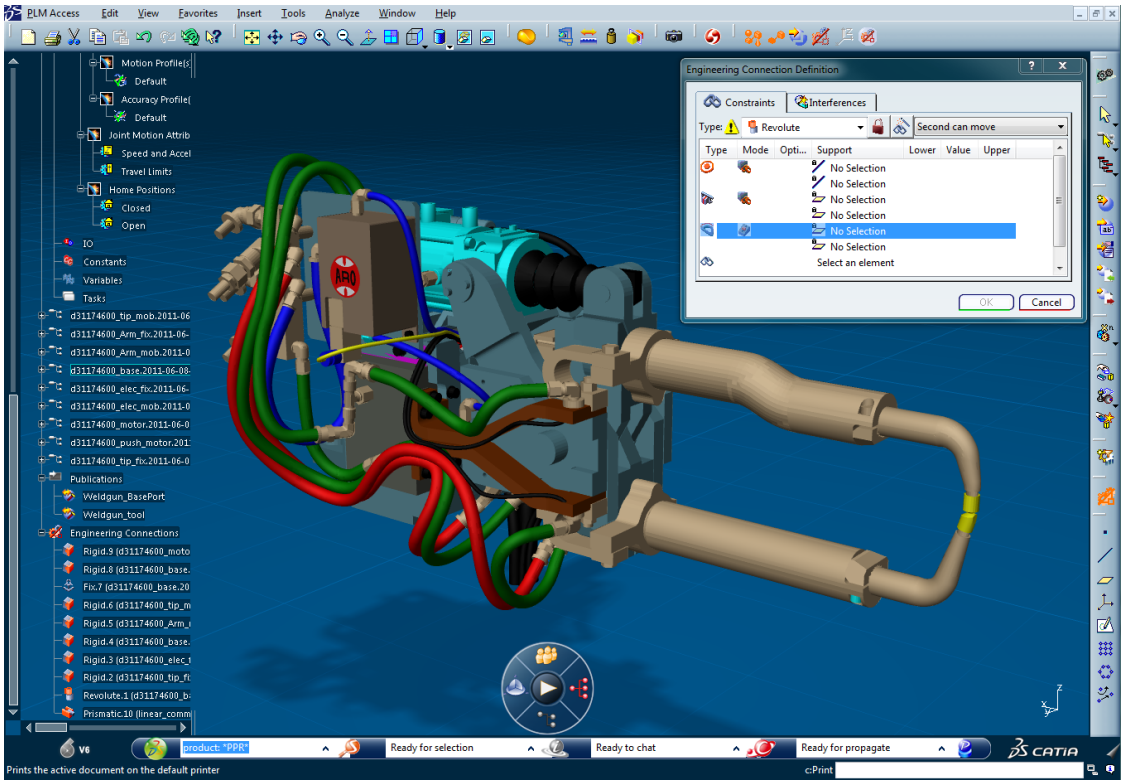


Figure 2.1: *A figure showing the "Engineering Connection Definition" dialog. Also a list of created engineering connections for the weld gun tool can be seen at the bottom of the tree.*

2.3 Resource Hierarchy

Within DELMIA "Product" is a pivotal term. A Product is a root element containing other elements such as 3D model parts. These parts can for instance be .cgr files or any other supported CATIA model formats. In the Resource Definition & Layout workbench there is a function which can transform existing Products into a "Resource". The virtual cell consists of a number of Resources arranged in different levels of the tree and with different purposes. DELMIA has a number of different resource types which have specific properties and functions. The ones used in this project are explained in Appendix A. Each of the two spider assemblies are considered separate devices, together with all the clamps, centering pins, sensors and the metal plate they are positioned on. They should be organized under one common parent resource. The logic controllers that control the tool devices which the turntable consists of were placed in the top level of the tree, in the organizational resource of the cell. An area resource called "13-95-050-Turntable" contains a Tool device which is a model of the turntable itself. It also includes the rotary table that holds both spider assemblies. Lastly, it contains two area resources for

the spider assemblies which in turn contain all the pins, clamps, sensors and the metal plate geometries. See Figure 2.2.

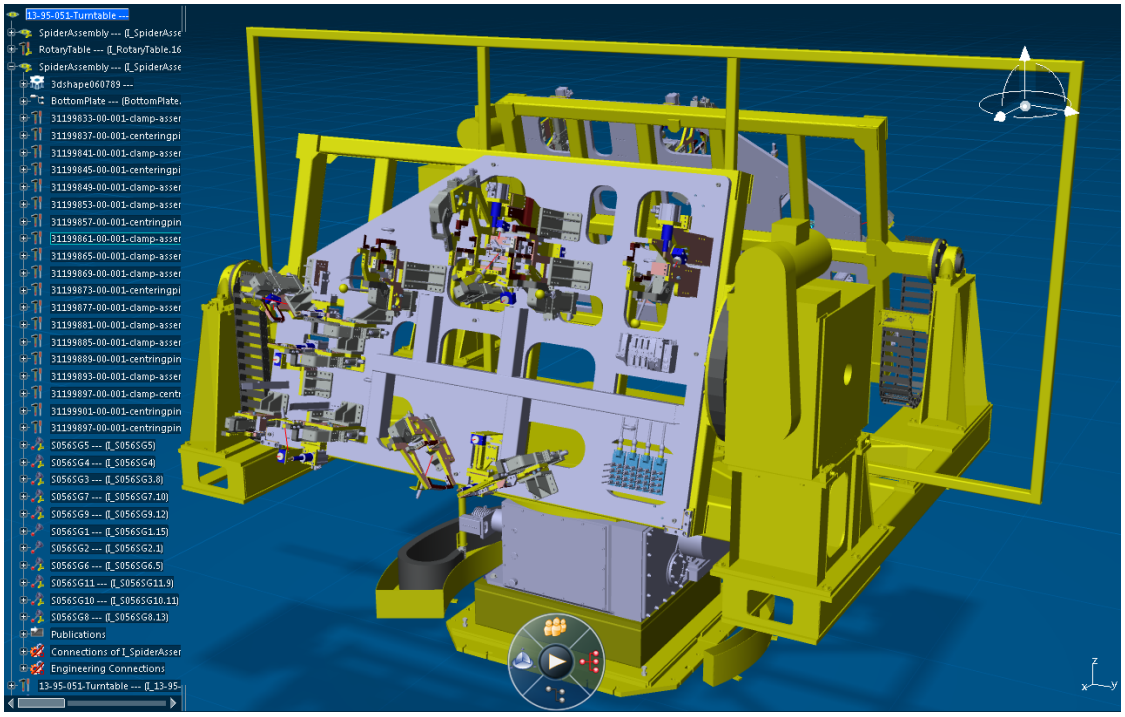


Figure 2.2: *The hierarchical organization of the turntable. An area resource is at the top level, containing the rotary table, the turntable and the two spider assemblies which in turn contain the clamps, guiding pins and sensors.*

The approach for the robots differs; an organizational resource holds all related, the robot resource, the tool, the control device and the logics resources. The whole organizational resource appears as an entity as seen by the cell. The I/O ports are connected directly to this resource, unlike the components of the turntable, where the I/Os are handled by the control devices.

2.4 Attachments & Mechanical Ports

In order to enable different objects to move together attachments between them must be created. This is done by adding or selecting an existing axis system on each of them and defining attachment ports. Depending on which of the objects should move with the other, one will be considered ”Master” and the other one ”Slave”. Both of the robots are masters while the gripper and weld gun are slaves. The attachment port (also mechanical port) type should be ”Mount” for the Master object, and ”Base” for the Slave object. Robots imported from the DELMIA robot library automatically have attachment ports defined at the correct position. When

the attachment operation is completed it is possible to use the snap option to put the two objects together, with respect to their respective axis origins.

The same approach was used for the turntable. The two spider assemblies are attached to the rotary table, which in turn is attached to the mobile part of the turntable. This means the spider assemblies have base ports, the rotary table has both mount and base and the turntable has a mount port.

2.5 Simulation Logic

Simulation Logic is created for devices in DELMIA in order to mimic the behavior of their real world counterparts. Actuators such as air valves and servos are imitated through SFC programs together with device tasks. All devices in the project have a sequence of possible actions depending on the input. Also outputs from internal sensors can be modeled in this manner. An example can be seen in Figure 2.3. It is of importance to distinguish this type of virtual logics from actual logics which exist in the form of PLC programs. The logics are added to the associated control device in the "System & Device Designer" workbench. A program which controls the whole cell can be implemented in the Logic controller, which is useful for testing the behavior of the devices before connecting the actual PLC.

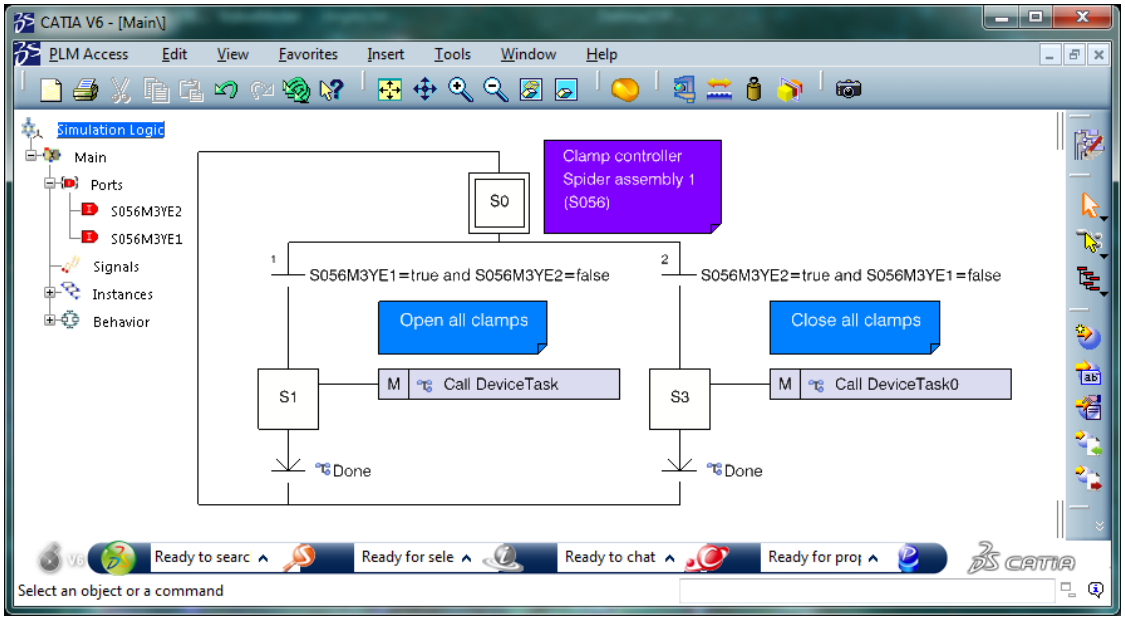


Figure 2.3: *Figure showing how PLC signals control the air valves that open and close the clamps of the spider assembly. The macro "Call DeviceTask" is a call for the task defined in the clamp controller to open the clamps.*

2.6 Discussion

The conclusion is that the provided 3D models from different simulation software are compatible with DELMIA V6 and can be imported and used for simulation. The .CATProduct, .CATPart and .cgr CATIA V5 models proved to be the formats which worked the best. A small number of solids and surfaces in the models disappear when migrating files from CATIA V4 to V5/V6. The same problem appears when handling .stp files. The available spider assembly model is for the other side of the production line, which assembles parts for the right side of the car. Therefore the whole spider assembly had to be mirrored. This step proved to be a problematic and workload-heavy procedure. Some details could not be mirrored and the color information was lost in the process. Also it was found to be simpler to first prepare all clamps, pins and the turntable itself in DELMIA V5, as it is simpler to handle .cgr files. With the function "Frame of interest" an axis system can be added at an arbitrary position, which is then used to create kinematics. This function appears to be removed from DELMIA V6.

The .step file format is a graphical model representation which generates large files. It is time consuming and requires good hardware to import these files into DELMIA. When imported and saved through DELMIA as the native .cgr format the file size becomes significantly smaller. To further optimize the 3D models a function in DELMIA V5 called Simplify can be used, which reduces the number of triangles used in the polygonal mesh. An alternative to .step files is .jt which also can be exported from Process Simulate. The DELMIA version used for the project could not handle this format, to support it an extension is required. It would simplify the data collection, as sometimes the .step files were too large to open, and also required large pen drives to be moved from VCC.

Moreover there is an issue within DELMIA V5 and V6 which makes it difficult to work with some .cgr files. The compass has the ability to snap to selected objects, appearing at the geometry of the model. By editing the coordinates of the compass the object can be moved and translated freely. However it does not snap properly for .cgr files, as the compass always appears in the origin. This issue makes it difficult to place the 3D models, and forces the use of Snap resource. The compass can be placed manually on the geometry, but when editing the coordinates it will not move the desired geometry.

A flow of work pieces through a production cell model can be implemented by adding sources which generate products at a given position, and sinks which remove the products. Generated work pieces only exist transiently during simulation, and are not considered actual PLM products [9]. To detect these products so called simulation sensors are used, which use beams to sense them as they pass by. In a similar manner, to enable transportation of the products throughout the cell, grab sensors must be utilized. However grab sensors do not work properly in the V6R2012X and earlier versions, which means sources and sinks cannot be used to generate and dispatch products. As an alternative a function called "Pick and

drop" is used. It causes preselected products to move along with a device which calls the function. Only one cycle can be run in the simulation since it is not possible to generate new products.

In the final version of the model of the cell only one of the 11 sensors is used per turntable. Since work pieces are not generated, sensors require the sensing mode which detects all geometries rather than just work pieces. This detection mode reduces the simulation performance since it requires heavy computations constantly.

There are a number of variables which can be set to make the simulation more accurate. All movements related to the devices of the turntable have default values for speed and acceleration. No timings were acquired from VCC.

3 Importing Robot Geometry and Programs

DELMIA has a library containing models of the most commonly used robots from different manufacturers. Once a library is imported to the project all the robots from the selected manufacturer will be available for use. These models already have all the kinematics, joints and attachment ports defined when imported. The production cell used for this project contains two different robots from ABB. One of the robots is an IRC5 6640ID and has a weld gun tool attached. It is a servo controlled scissor type spot weld gun from ARO. The other one is an S4Cplus 6600 equipped with one of two gripper tools which have five clamps used to move the work pieces, the tool change occurs automatically.

3.1 Robot Programs

In DELMIA it is possible to import robot programs and obtain the behavior of the actual robot for the model. This feature saves time and work as it automatically creates the tasks and points in the space, also known as tags. Considering the fact that different robot manufacturers use different programming languages there are a number of translators for the respective languages available in DELMIA. Copies of the .mod files used in the robots were provided by Volvo Car Corporation and the unrelated procedures for other products were removed from the files. Programs for the tools are located in separate files. Also, a number of programs for the communication between the robots and the PLC were included. The tools in the "Robot Offline Programming" workbench allow the user to upload and translate a text file containing the robot program to use for the desired robot. ABB robot programs are written in RAPID, which is a C based programming language [14]. When the translation is finished a report window displays the notices, warnings and errors that occurred in the process (See Figure 3.1). The Task Options tab allows attaching tags to parts by selecting one part per object frame [9].

The RAPID translator can comprehend all robot targets which are arrays of constant values and save them as tags. The different types of movements such as linear movements (MoveL) and joint movements (MoveJ) between the robot targets are used to create "Motion Activities" in DELMIA. Different properties of the movements such as tool profile, motion profile, accuracy profile and object profile are automatically assigned. Together, the whole cycle of motion activities forms a "Task". SpotL and SpotJ are in a similar manner translated into spot weld operations. However the RAPID translator does not understand other specific commands such as "Reset" and "WaitSignal", which have to be recreated manually if considered to be required for the simulation.

There are certain limitations the version of the translator used for the project. Local variables are not working in the V6R2012 version. The workaround is to replace "LOCAL CONST" with "CONST" and "LOCAL PERS" with "PERS".

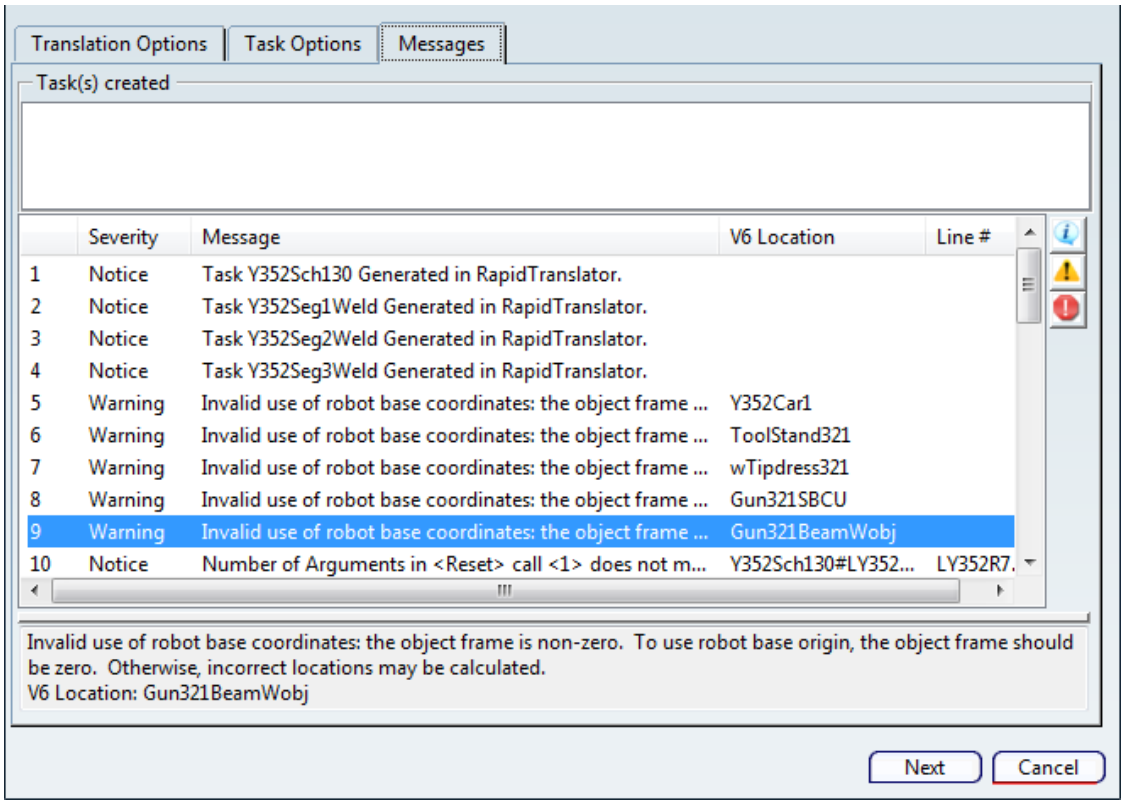


Figure 3.1: *A figure showing the "Import robot program" feature in DELMIA V6. The selected tab shows a report of the robot program translation, and includes descriptions for the warnings.*

Moreover, the spot welding data in the welding robot program has to be edited. Standard S4C/S4C+/C5 syntax supports four arguments whereas the robot programs supplied by Volvo contain five arguments[1]. One solution is to reduce the number of arguments to four, or to customize the RAPID translator to support the fifth argument. Another limitation is that the two separate .mod files containing the gripper and weld gun programs cannot be uploaded in the current version. There are a few reasons for this:

1. The robot program files contain Select/Case syntax which is not supported by the RAPID translator.

2. Contains Arrays of Strings, which are not supported.

3. Identifiers larger than 16 characters are not supported by the RAPID translator.

[1]Hairon Anthony 2011, email, 23 September, <anthony.hairon@3ds.com>

As a result, the specific commands for the gripper cannot be imported. Thus the opening and closing of its clamps has to be recreated manually. In the actual cell, the PLC has no control over the gripper, it is entirely controlled by the robot. It should not be defined as the robots auxiliary axis. The two tasks are created in the grippers motion controller. To enable the robot to control the opening and closing instructions it is possible to call them as services within the robot tasks. Closing the clamps of the gripper will not actually grab the work pieces, to enable them to move together with the gripper the "Grab" command must be initiated. Once the clamps are opened, the "Release" command makes the parts remain at the current position. No specific information was received about the clamp opening sequence, studying the behavior tells that one clamp is opened before the gripper approaches the pallet, and once the parts are in place and fixed by the pallet, the remaining clamps are opened. This behavior is modeled according to Figure 3.2 below.

To recreate the communication interface between the robots and the PLC both ends are studied. A routine called DoOrder is used for executing orders from the PLC. The PLC sends an order with a specific number based on which product is processed, which in the robot program corresponds to an entry in a list of CASEs. For this project only one procedure is used for each robot, the one which processes the V60 model. Also the robots communicate with the PLC to initiate a number of operations, like ordering opening of clamps and allocating zones. This part of the communication is carried out by sending "Locksignals". The signals consist of numbers of the data type Word, for each request there is a specific number. When a request is sent to the PLC it answers with a signal of the same value once it is accepted. The translator does not have the ability to recreate such tasks. It can be recreated by writing required procedures in SFC code. The exchange of Locksignals can be modeled by creating a step with an output containing the value of the Locksignal and a transition with the corresponding input as a condition.

3.2 Discussion

Unfortunately the RAPID translator is limited to a certain extent and cannot handle all the contents of the robot programs. Without introducing a few changes to the programs the compilation and upload is not possible. To conclude, the translator merely handles the most crucial part, namely the creation of tasks with movements and weld spot points. The parts which cannot be uploaded must be added manually. If this procedure is to be done for a large number of projects it would be more efficient to extend the support of the RAPID translator. The two main programs used for the robots in the production cell could be uploaded directly to DELMIA after introducing a small number of changes to the .mod files containing the programs. As previously mentioned the two tool programs cannot be uploaded, hence the home positions (like HomeGun321 or HomeGripp412) are

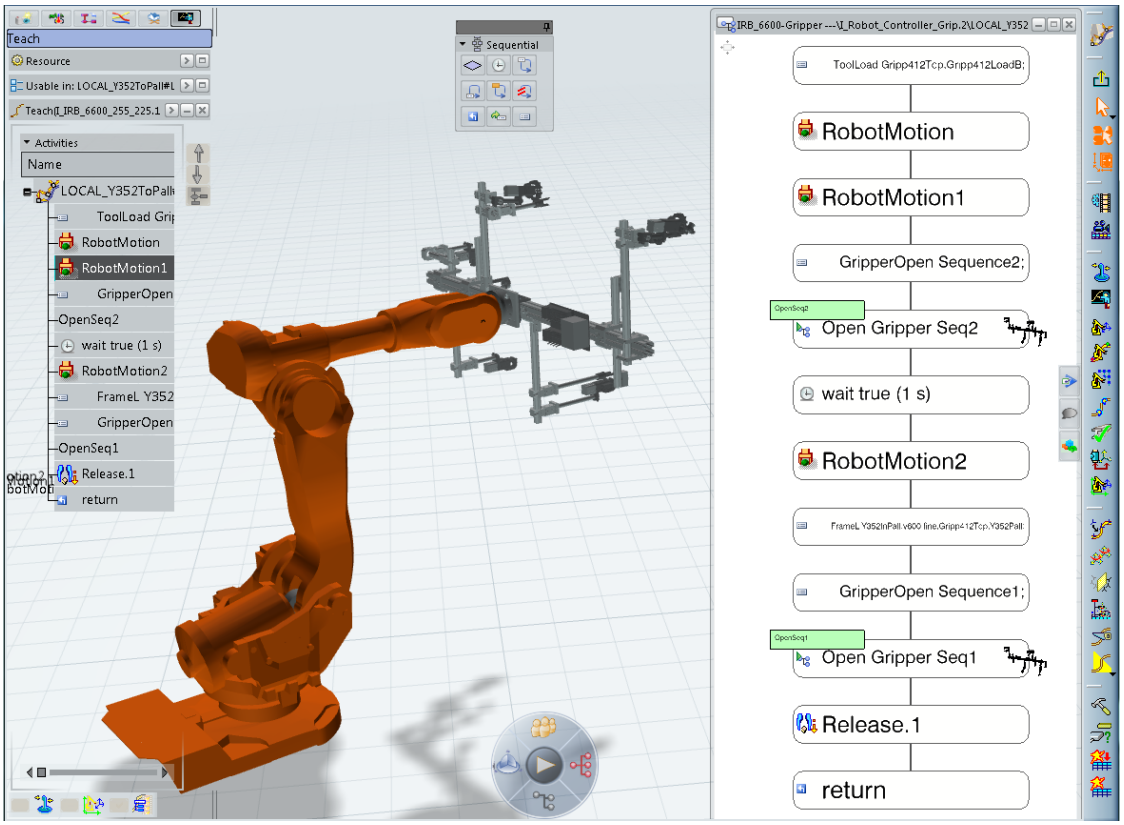


Figure 3.2: *Figure showing one of the uploaded tasks in the "Teach panel", both with commands which are successfully translated by the RAPID translator and the ones which are not, such as ToolLoad and GripperOpen. "Open Gripper Seq2" is one of the added services which are called by the robot, and "Release.1" represents the releasing of the parts. A timer has been added to let the clamp open before proceeding with next instruction.*

not imported. The target data for home positions needs to be provided in the main program, by copying and pasting it from the respective tool program. The two tool programs contain tasks such as dressing of the weld gun, service of tools and weld testing. Because of the scope of the project most of these tasks are not necessary to recreate. However, the necessary ones can be created manually in DELMIA's Device Task Design workbench. Before uploading the robot programs a S4C controller profile should be uploaded and applied to the two robots in DELMIA. In its settings the WorldCoords parameter should be set to Boolean value "false". The purpose of this is to give correct tag positions in space.

4 PLC

Programmable logic controller (PLC) or programmable controller is an industrial computer used for automation of electromechanical processes. PLCs are designed for multiple input and output arrangements, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. PLCs have a critical role in virtual commissioning projects. One of the main objectives of virtual commissioning is to validate the performance and operation of the system, which is represented as a 3D model in the simulation software, against the actual PLC.

4.1 Hardware Preparation

The required hardware for this project is a Mitsubishi PLC. A Q12HCPU PLC with 124 k step program memory which was provided by Volvo Car Corporation was used in this project. One of the important features which the Q series of the Mitsubishi PLCs supports, compared with other Mitsubishi PLC models, is the possibility to force inputs/outputs, i.e. within the programming environment (GX IEC Developer) the automation engineer can change the value of variables when in online mode. This property allows to see the operation of the PLC in the online mode and to force variables with the desired value for troubleshooting purposes. The hardware configuration used for this project is as follow:

- Base unit
- Power supply module
- Q12HCPU Mitsubishi PLC (High-Performance CPU family)
- QJ71E71 Ethernet interface module (Network module)

Figure 4.1 illustrates the Mitsubishi PLC system structure used in this project, except that no physical I/O modules are used in the communication between the PLC and the 3D model.

4.2 Software Preparation

The programming environment for the Mitsubishi PLC which is used in this project is GX IEC Developer 7.04 software. The PLC program which was provided by Volvo Car was a large program (approximately 15,000 I/Os) which controls multiple cells and the transportation lines between them, that carry the car body parts through the production line. Volvo Car Corporation provided a PLC program which was simplified as much as possible, however even this version was considerably comprehensive.

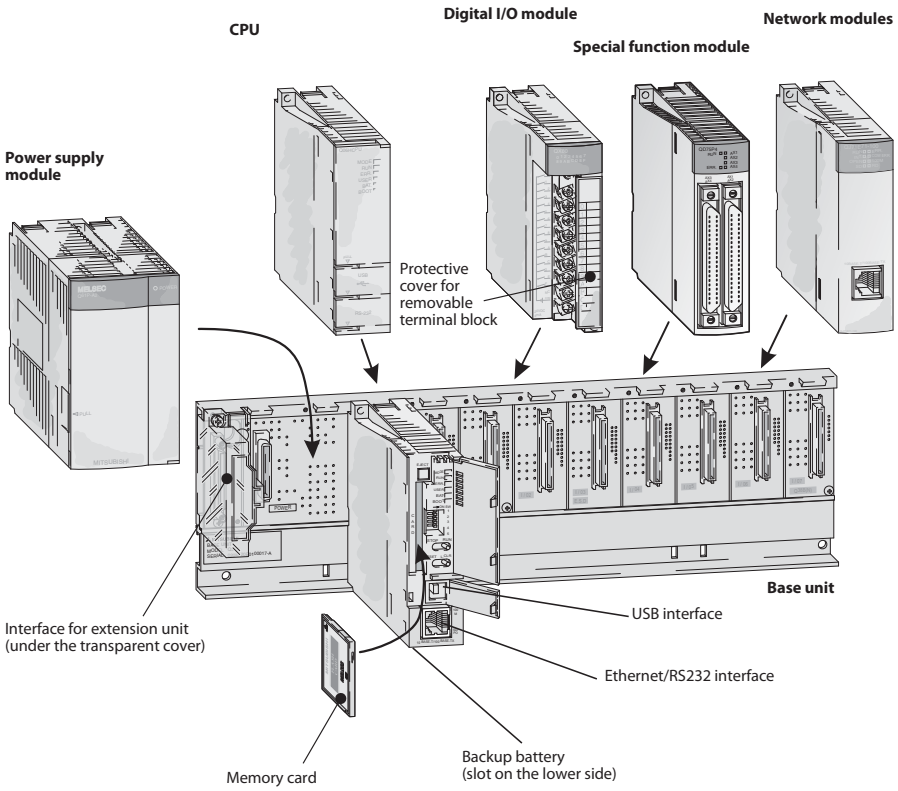


Figure 4.1: *Mitsubishi PLC system structure*

The PLC program consists of 51 Program Organization Units (POU) which may contain Ladder or SFC programs and have in total 39132 steps, i.e. about 39 kilo steps. The PLC program was reviewed completely and some parts of the program which were unrelated to this project were bypassed and removed from the main program. In most cases it was necessary to modify the PLC program in order to override interlock signals between the cell used in the project and other cells. For instance, in the PLC program there was a POU for communicating with the two Profibus modules which were used in the production line at the Volvo Car factory. Since these Profibus modules were not required, the parts of the PLC program which were related to the communication with these two Profibus modules had to be bypassed otherwise the program did not run. Figure 4.2 shows the Task_Pool of the PLC program in which the task DP1 is responsible for communication with the two Profibus modules.

In Figure 4.2, the task DP1 is bypassed by assigning a new variable PROFIBUS_ENABLE to its "Event" property and its value is set to FALSE. Therefore the task DP1 runs if and only if the variable PROFIBUS_ENABLE is set to be TRUE.

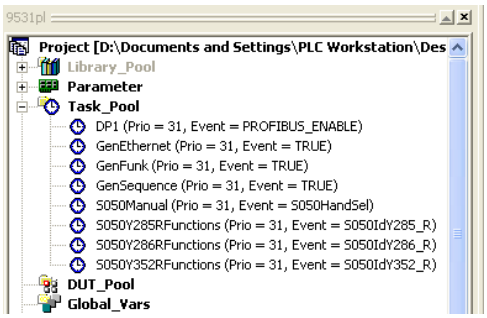


Figure 4.2: *Task pool of the PLC program*

The cell consists of two industrial ABB robots where each robot has its own program which handles the communication with the PLC. One of the important tasks in the PLC part was to find all signals in the PLC program which are needed to communicate with the robot programs. The electrical document generated with EPLAN Electric software, was used in order to find a clear point to point connection between the names that are used in 3D model files and the tag numbers in the PLC program.

In order to acquire the information from the PLC program it was necessary to extract the flowchart of the program for each task. For example Figure 4.3 shows the program flow for fixtures S056 and S057. These are the two spider assemblies, that are mounted on the turntable. In this flowchart the interconnecting signals which are used for the communication with the other cells for safety are not shown.

This EPLAN file contains following information about the workcell:

- Installation drawings
- Circuit diagram
- Cable list
- Input/Output list
- Termination diagram

The EPLAN document, which was the only electrical reference in this project, was used to find the connection between the names of sensors and the actuators in the 3D model and the tag numbers and the related signals in the PLC program.

4.3 OPC Configuration

The OPC communication protocol was used for connecting the Mitsubishi PLC to the 3D model in DELMIA V6. OPC which stands for OLE (Object Linking and Embedding) for Process Control is the protocol which specifies the communication

of real-time plant data between control devices from different manufacturers [5]. There are multiple OPC server softwares which are almost the same in operation but each one has its own performance and capabilities. Multiple OPC servers were examined for this project such as Matrikon OPC server and KEPServerEX, but finally the Beijer OPC server was chosen. The Beijer OPC server is the most user-friendly OPC server of the ones tested and is the least complex. Volvo Car Corporation provided the Beijer OPC server for this project. The server connects to the PLC, gathers required data and transforms this data to a standard OPC format. Then DELMIA V6 can connect to it as an OPC client. In DELMIA, SFC code was added which is used to make the connection between I/Os of the sensors and actuators in the 3D model and the inputs/outputs of the actual Mitsubishi PLC. These signals are sent to DELMIA V6 through the OPC communication protocol. The OPC communication protocol plays an important role in this project. It is the communication bridge between the actual PLC and DELMIA V6 which allows them share data mutually. The OPC server is installed on the computer which DELMIA V6 is running on; therefore some potential network restrictions and firewall problems are bypassed. However in general, the OPC server can be installed on another computer as well. Figure 4.4 depicts the general configuration of the system. The software on each workstation is as follows:

- DELMIA workstation: DELMIA V6 (as an OPC client) - Beijer OPC server
- PLC workstation: GX IEC Developer (for PLC programming)

As can be seen in Figure 4.4, there is a serial connection between the Mitsubishi PLC and the PLC workstation. This USB connection is used for programming the PLC, monitoring and debugging the program.

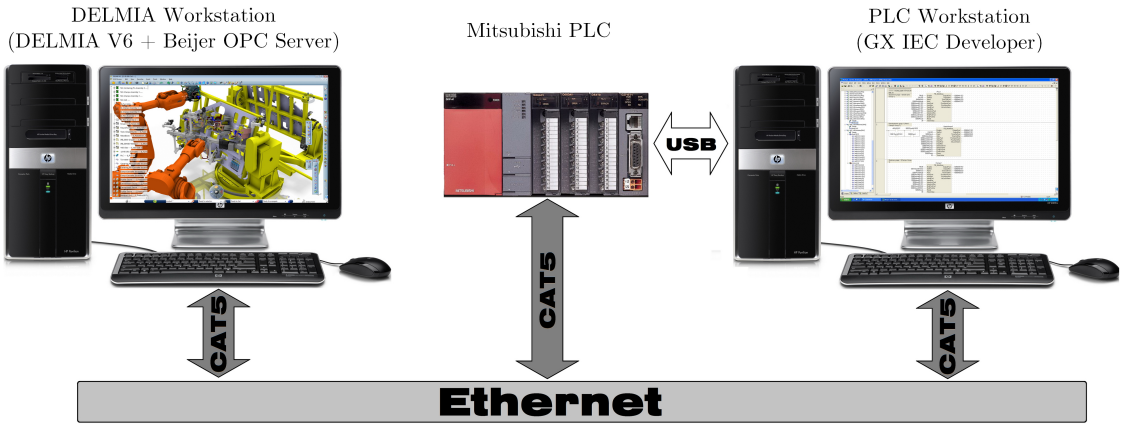


Figure 4.4: *General hardware configuration of the system*

Required I/Os were defined on the OPC server and are read from the PLC while running. These I/Os are categorized in a number of different groups in the

Beijer OPC server software. Table 4.1 shows a part of the Input/Output mapping list which is prepared for the communication between the PLC and DELMIA. This table which contains about 110 I/Os has the required information which is necessary for communication through the OPC server. The signals in this table are mapped into the OPC server and are accessible by DELMIA which acts as an OPC client.

Table 4.1: *A sample part of Input/Output Mapping List*

I/O (PLC Side)	Name (PLC Side)	MIT-Add.	Type	Object No (DELMIA)	Action	Group
Output	S056M3YE2	Y1384	BOOL	–	Close clamps	Spider Assem. 56
Output	S056M3YE1	Y1385	BOOL	–	Open clamps	Spider Assem. 56
Output	S057M3YE2	Y13D4	BOOL	–	Close clamps	Spider Assem. 57
Output	S057M3YE1	Y13D5	BOOL	–	Open clamps	Spider Assem. 57
Input	S056SG11	X13B9	BOOL	31199853	Photocell sensor	Spider Assem. 56
Input	S056SG6	X13B6	BOOL	31199841	Photocell sensor	Spider Assem. 56
Input	S056SG3	X13B2	BOOL	31199853	Proximity sensor	Spider Assem. 56
Input	S056SG4	X13B3	BOOL	31199845	Proximity sensor	Spider Assem. 56
Input	S056SG5	X13B4	BOOL	31199837	Proximity sensor	Spider Assem. 56
Input	S056SG8	X13BA	BOOL	31199877	Photocell sensor	Spider Assem. 56
Input	S056SG10	X13B7	BOOL	31199885	Photocell sensor	Spider Assem. 56
Input	S056SG7	X13B8	BOOL	31199877	Proximity sensor	Spider Assem. 56
Input	S056SG9	X13B5	BOOL	31199889	Proximity sensor	Spider Assem. 56
Input	S056SG1	X13B0	BOOL	31199901	Proximity sensor	Spider Assem. 56
Input	S056SG2	X13B1	BOOL	31199897	Proximity sensor	Spider Assem. 56

Figure 4.5 shows a snapshot from the running OPC server. It can be seen that there are six main groups of introduced tag numbers for the OPC server. One group which is called SystemVariables, is automatically generated when a new controller is defined. In Figure 4.5, the VolvoPLC is the chosen name for the controller. These I/Os were introduced to the OPC server as tag numbers with their address. For introducing an I/O to the OPC server some necessary information is required such as: name, address, data type and poll interval (how often the tag value is to be updated). The name of defined tag numbers should comply with the names in the PLC program and are best chosen to be something meaningful, however for further clarity the chosen names of the tag numbers in this project are exactly the same as the ones of the PLC variables. The address of the I/O can be in either IEC or MIT (Mitsubishi addressing procedure) formats. The GX IEC Developer can recognize both the IEC and MIT formats for addressing of variables. The required tag numbers which are defined in the OPC server can be accessed from any OPC client which can detect the running OPC server.

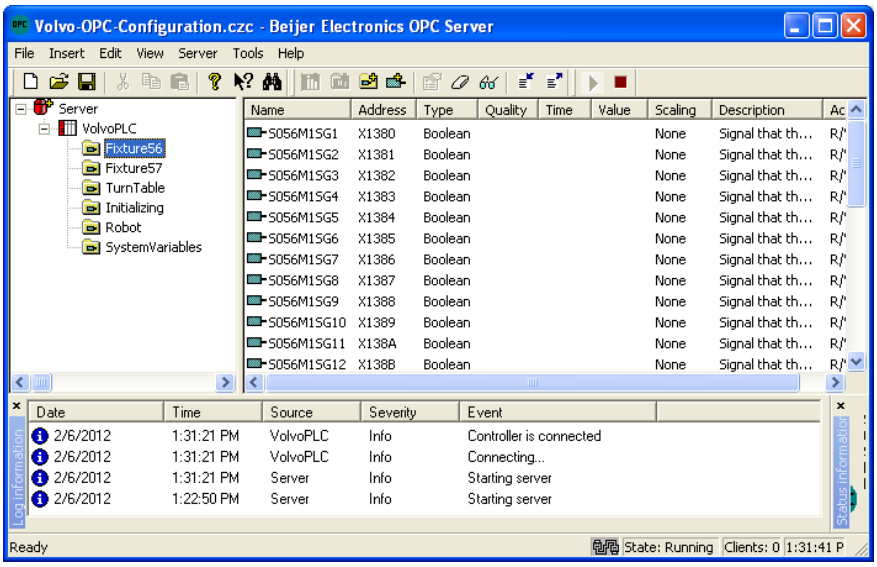


Figure 4.5: *Running OPC server (list of signals and their information for the fixture 56 are shown)*

4.4 Discussion

The general concept of virtual commissioning in this project is validated in terms of communication with an actual PLC. DELMIA V6 is connected to the actual Mitsubishi PLC. Although some modifications were needed in the PLC program in order to work properly. Writing the PLC program while designing the product and production system reduces the amount of code debugging (i.e. backtracking signals in the PLC program).

There are a number of POUs in the PLC program which are unrelated to the station 13-95-050 but unfortunately could not be completely separated from this station since it has a number of interconnection signals with station 13-95-050. Many of the backtracking tasks are done to follow these interlocking signals and to bypass them.

None of the function blocks in the Library_Pool of the PLC program could be edited, these function blocks caused some difficulties for communication between the PLC program and the robot programs. These function blocks belong to VolvoLib2_0 library.

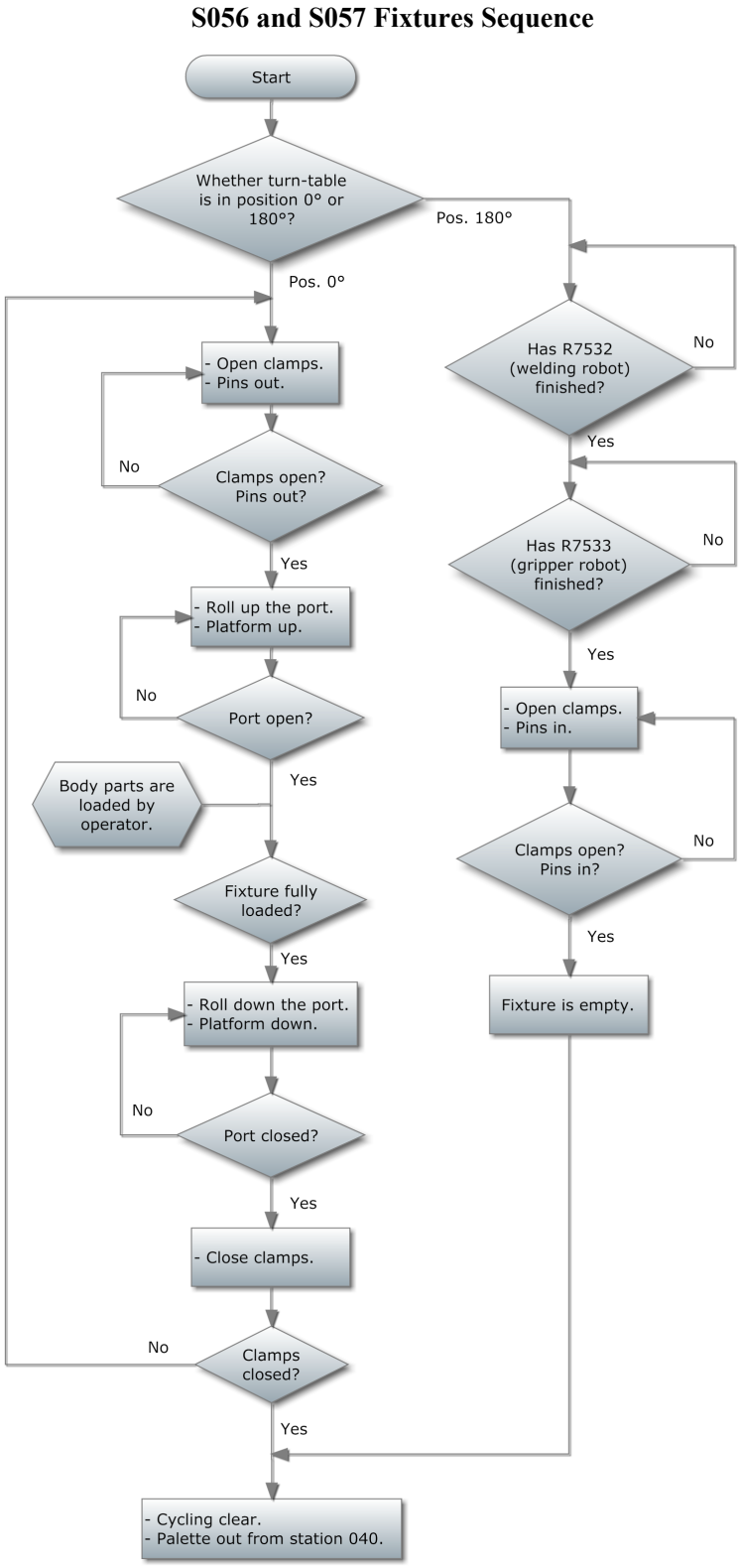


Figure 4.3: *Program flowchart for fixtures S056 and S057*

5 Final Simulation

With the 3D model ready to simulate, the OPC server and PLC configured, the two major parts of the projects are then linked together. In DELMIA, the previously used internal logics ("software PLC") are replaced with the external control. The external control is connected with the OPC communication protocol and receives all available tags from the server (see Figure 5.1). The tags appear as I/Os of the external control block. As previously mentioned the actual PLC has a number of variables which need to be initialized to certain values before the program can be executed successfully. This can be achieved by creating simulation logics inside the internal control block and internally connecting the I/Os of this block with the External control block. A 2D HMI[2] is added which represents the control panel the operator interacts with in the actual station. It can be used to trigger the cycle once all four workpieces are in place in the spider assembly.

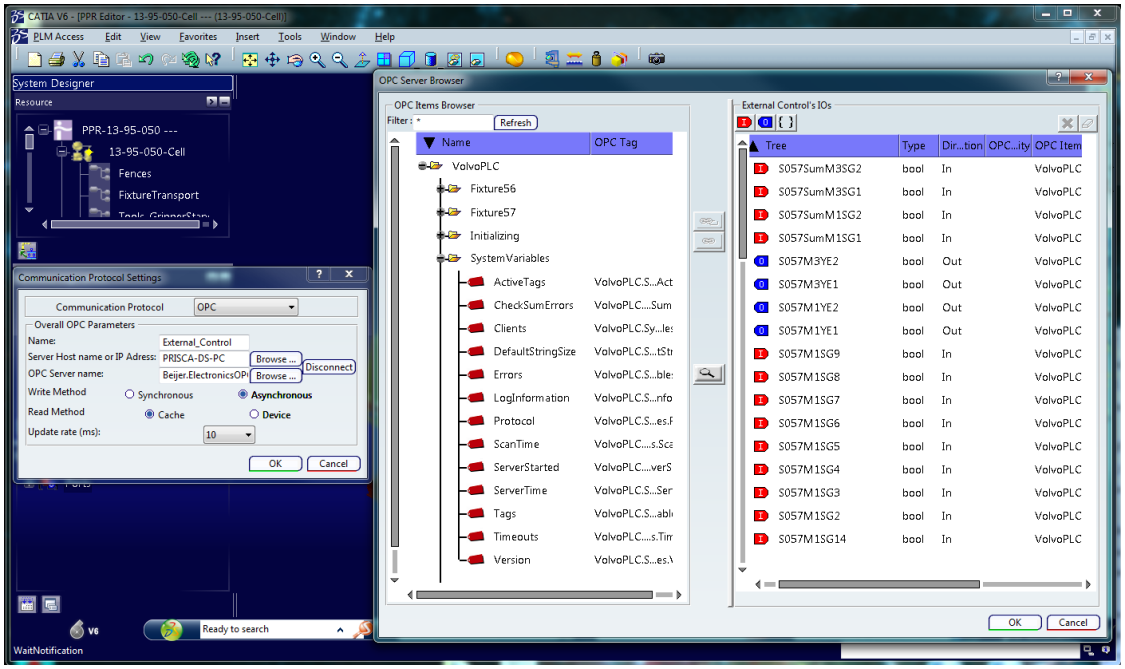


Figure 5.1: *DELMIA's external control interface. To the left, the smaller window shows the settings for connections. To the right, the larger window shows the OPC server browser, where the mapping between OPC tags and DELMIA I/Os is done. The fields to the left show the different available tags and the field to the right shows the I/Os of the external control block.*

In the device creation phase all I/Os of the devices are defined. These I/Os and the connections of the whole DELMIA cell model can be seen in Figure 5.2.

[2]Human-Machine Interface

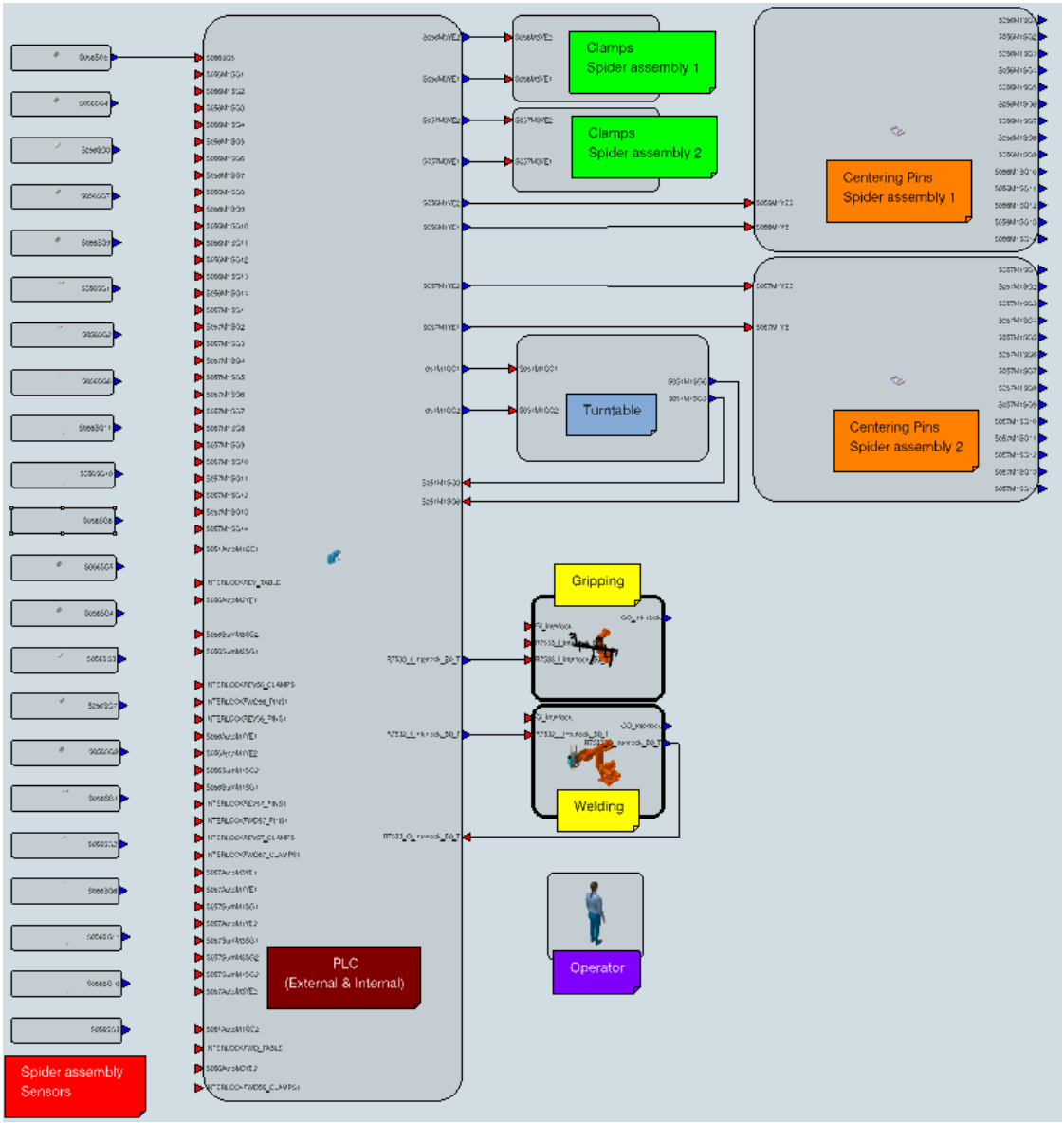


Figure 5.2: *Figure showing an example of the connections of the different devices in the production cell and their respective I/Os. Red arrows represent inputs while blue arrows are outputs.*

There are a few simulation options available which alter how the simulation runs. "Simulate in Real Time" can be selected, which will lead to continuous update of the step size. Slower simulation with respect to real time results in larger step size, and vice versa. This option can cause varying simulation performance depending on how heavy computations are at the given moment.

6 Conclusion & Future work

In this project, necessary steps for preparing available resources for virtual commissioning are introduced. Dassault Systèmes offers a complete digital manufacturing solution, from the early product and production design to the final commissioning. The V6 version of the software integrates ENOVIA which allows everyone assigned to the project to have access to the files. Any changes done to the files are propagated to the server. Comparing to V5, the new design aims to simplify the use of the software. An online documentation is available to help users become familiar with the different tools. CATIA/DELMIA/ENOVIA are integrated into a seamless environment, meaning that everything is done in one window without having to export and re-open in other editors. Instead, workbenches are used which contain tools specific for the desired purpose. In fact, DELMIA is an extension of CATIA. This allows an effortless transition of CATIA resources to DELMIA. It is clear that a big effort has been put in to change the way of working with DELMIA. The new version of the software (V6) feels redone, it is aesthetically more pleasing and some functionality is more straightforward compared to V5. However, some functionality was missing in these versions, and therefore, version V5 was preferred for certain tasks. The exportation of .cgr files was done in V5 as the first release of V6 used for the project lacked the possibility to export this format. The DELMIA V6 documentation is slightly lacking of necessary information and is pedagogical. It often describes well how to do most operations but it is difficult to know in which situations they can be used. Some chapters of the documentation are thorough while some need revising. For users new to DELMIA it can be hard to navigate and know the correct procedures. The most efficient way of becoming familiar with the software is to have personal training and to follow tutorials.

Regarding the essential robot part of the virtual cell, it is possible to improve the RAPID translator to include non-standard functions that are implemented in the robot programs. For this project the translator was not edited and instead part of the robot tasks are added manually in DELMIA. Almost all of the robot functionality which was required for this project is successfully translated by DELMIA. With a wider scope including tasks such as change and service of tools a more extensive amount of manual work would be required.

As for the PLC part, it was found that it is possible to simulate a cell using DELMIA V6 and the given data. Though it does not work instantaneously, many simplifications were done in the model, robot programs and the PLC program. To introduce a true virtual commissioning all stations including their functions needs to be implemented. Many blocks in the PLC program are dependent on the signals of other stations. The PLC code should be written by automation engineers from the early design steps in parallel with the creation of 3D models in order to achieve less code debugging.

To conclude the virtual commissioning part, the final cell is not an entirely

trustworthy copy of the actual cell; there are more steps needed to take which will increase the authenticity of the model, such as introducing collision detection, speed and acceleration of clamps, implementing mean-time-to-failure, etc. No timing information was available hence default values were used in all cases except for the robot movements. The option for the real time simulation could not be used with the hardware supplied for the project. The computations required were too heavy which leads to a simulation approximately three times slower than real time. A message informs that the simulation is unable to reach real time.

For future work, there are several distinctive paths that could be taken. One path can be improving the current work to include all three variants of the car models, and on making it more realistic by adding correct timings and collision detection. Alternatively, more focus could be put into the RAPID translator. For instance it can be edited to extend its support of non-standard functionality as well as work with the functions which are not supported such as Select/Case. The communication between the robots and the PLC could be more elaborate by introducing interlocks.

References

[1] Digital Manufacturing and Production - DELMIA - Dassault Systèmes [online]. <www.3ds.com/products/delmia>, Accessed February 2012.

[2] Google SketchUp (2012) [online]. <sketchup.google.com>, Accessed February 2012.

[3] Is SAP PLM for Real? (April 2009) [online]. <www.scmfocus.com/servicepartsplanning/2009/04/21/is-sap-plm-for-real>, Accessed February 2012.

[4] NyTeknik, (February 2009) [online]. <www.nyteknik.se/nyheter/automation/article504438.ece>, Accessed February 2012.

[5] OLE for process control - Wikipedia, the free encyclopedia (2012) [online]. <en.wikipedia.org/wiki/OLE_for_process_control>, Accessed February 2012.

[6] PLM - CIMdata - Global Leader in PLM Consulting (2012) [online]. <www.cimdata.com/plm.html>, Accessed February 2012.

[7] Process Simulate - Siemens PLM Software (2012) [online]. <www.plm.automation.siemens.com/en_us/products/tecnomatix/assembly_planning/process_simulate>, Accessed February 2012.

[8] SAP - PLM Software — Product Lifecycle Management Solution [online]. <www.sap.com/solutions/business-suite/plm>, Accessed February 2012.

[9] V62012X User's Guides, Dassault Systèmes (2002-2012) [online]. <www.3ds.com/support/download-documentation/v6-users-guides/>, Accessed February 2012.

[10] Virtual Commissioning - DELMIA - Dassault Systèmes (2002-2012) [online]. <www.3ds.com/solutions/cross-industry-solutions/overview/digital-manufacturing/virtual-commissioning>, Accessed February 2012.

[11] What is PLM — PLM Technology Guide (2008) [online]. <www.plmtechnologyguide.com/site>, Accessed February 2012.

[12] R. et al. Drath. An evolutionary approach for the industrial introduction of virtual commissioning. *Emerging Technologies and Factory Automation*, pages 5–8, 15-18 September, Hamburg 2008.

[13] Pellicciari M. et al. Engineering method for adaptive manufacturing systems design. *Int. J Interact Des Manuf.*, 3:81–91, 24 March 2009.

[14] Akkan G. Shoaei M. Verification, Optimization and Synthesis of Sequences of Operations. May, Göteborg. Chalmers University of Technology. 2009.

[15] Kühn W. Digital Factory - Simulation Enhancing the Product and Production Engineering Process. Proceedings of the 2006 Winter Simulation Conference, 2006.

A Appendix

A.1 Resource types in DELMIA V6

Tool device – A resource which can contain logics, tasks and motions. All of the clamps and pins are defined as tool devices. Also the robot tools (gripper and weld gun) are Tool devices, as the attachments for a robot must be of this type of resource.

Organizational – This resource cannot contain logics of its own, but organizes other types of resources within it. The whole cell is an Organizational resource.

Area – Is a "transparent" resource meaning that upper level resources can access the other resources contained within. Has a similar purpose as the Organizational resource, however it lacks some of its properties, such as the possibility to have I/Os.

Sensor – A special type of resource which contains an arbitrary shape and a beam attached to it which can be set to detect workpieces or any other geometry passing the beam. Contains one output of Boolean type by default.

Robot – Robots which are imported in DELMIA from the robot library are automatically of this resource type.

Control device – This resource is used for describing the behavior of devices on the same level of the tree through an SFC (Sequence Function Chart) program. It can have I/Os and control one or more tool devices, also used for controlling robots.

Logic controller – A logic controller resource can like the control device contain SFC programs and I/O ports to connect to other resources. This makes it suitable to be used as a software PLC. In this project it was used for testing and verification purposes. The logic controller can also directly be exchanged for external control when the OPC server is connected.

Worker – A worker that handles a process of some kind. The resource contains a manikin model.

A.2 Manual for Running the Simulation in DELMIA V6

1. Start the CATIA V6R2012X application.

2. Select the "VCC-Final" project in the options.

3. In CATIA, enter a suitable search phrase, for instance 'PPR*', and click on the magnifying glass. Right-click on the search result "PPR-13-95-050", and select **Open with >Open advanced...** (see Figure A.1). Then make sure that **"With all representations"** and **"With expanded children from database"** are ticked and press **Open** (see Figure A.2).

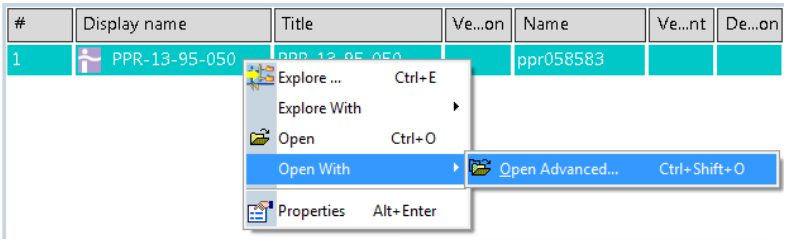


Figure A.1: *Opening the project*

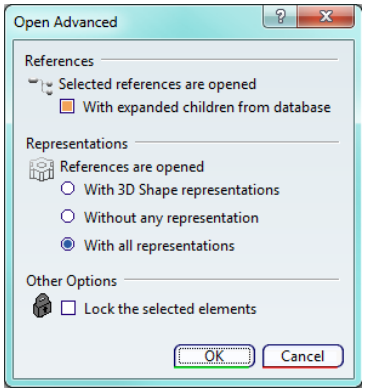


Figure A.2: *Opening the project; setting dialogue box*

4. Double-click on "13-95-050-Cell", in the tree. (If the tree does not appear press F3)

5. Now go to the **System & Device designer** workbench.

6. A tree containing all logics should open. Here expand the "13-95-050-Cell" branch, and scroll down to "PLC", select it.

7. Right-click on "External_Control" and select **Connection Protocol**. Browse and select the desired OPC Server. Also enter the name of the currently used computer (see Figure A.3).

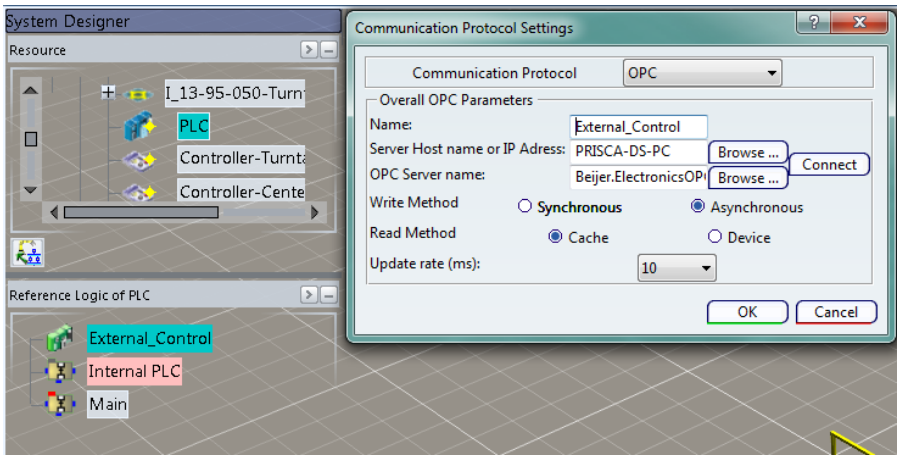


Figure A.3: *Communication with OPC server dialogue box*

8. To run the simulation, click on "13-95-050-Cell" in the top window, and then "Main" in the window below. Make sure that the OPC server is running, and press the "Play" button located in the middle of the compass, the simulation starts.

A.3 Manual for Running the PLC Program

This appendix describes the required steps to run the simulation in GX IEC Developer against DELMIA V6.

A.3.1 Setting up the PLC program

1. Run the GX IEC Developer program.

2. Open the project file by clicking on the yellow highlighted project folder which is mentioned in Figure A.4 (**9531pl**).

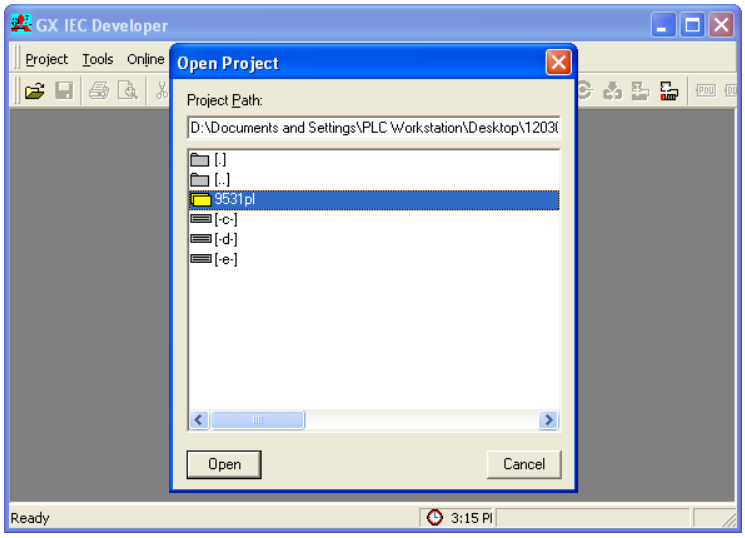


Figure A.4: *Opening the project dialogue box*

3. For applying modifications to the PLC program, its security level needs to be changed. However, for running the program and to view the signal status during a run, changing the security level is not necessary. For doing this, select **Change Security Level** in the **Project** menu. Security level is by default set to 1. Set the **Security Level** to 7 and leave the **Password** field empty and press **OK**.

4. The available PLC program is built and has already been downloaded to the PLC CPU. If any modifications are made to the PLC program it needs to be rebuilt and transferred to the PLC CPU before running the program again. In order to build and download the PLC program into the PLC CPU, perform the following steps otherwise jump to the next section (Monitoring Mode).

5. Select **Rebuild all** from the **Project** menu or simply use the **Shift+Alt+C** keyboard shortcut. Then the PLC program will be built again. The message which is shown in Figure A.5 will appear after building the program with zero errors.

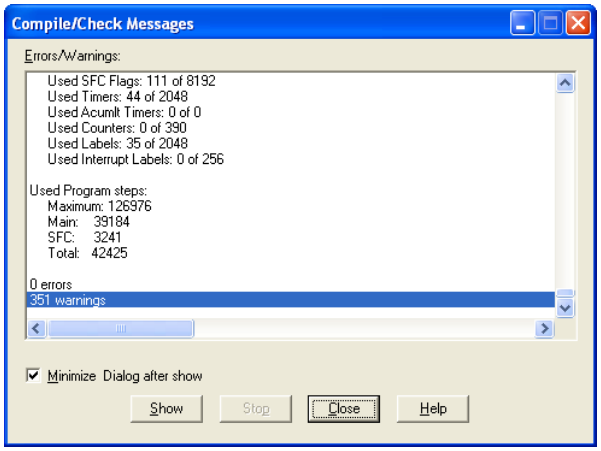


Figure A.5: *Building the project message box, this message box will appear after rebuilding the project*

6. The built program needs to be transferred to the PLC CPU before running the simulation. In order to do this select **Transfer** from the **Project** menu, then click on the **Download to PLC** or use the **Ctrl+Alt+W** keyboard shortcut. Following dialogue box will appear (see Figure A.6).

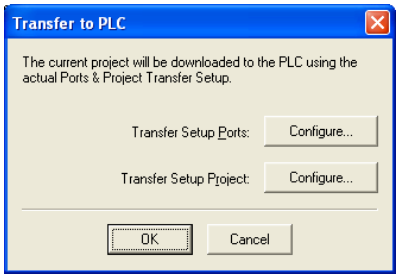


Figure A.6: *Transfer to PLC dialogue box settings*

7. Select the **Configure** button for checking the **Transfer Setup Ports**. Then make sure that the settings are the same as in Figure A.7. The USB connection is chosen for the communication between the PLC CPU and GX IEC Developer. After checking the transfer setup settings press **OK**.

8. Select the **Configure** button for checking the **Transfer Setup Projects** from the **Transfer to PLC** dialogue box. Then make sure that the settings

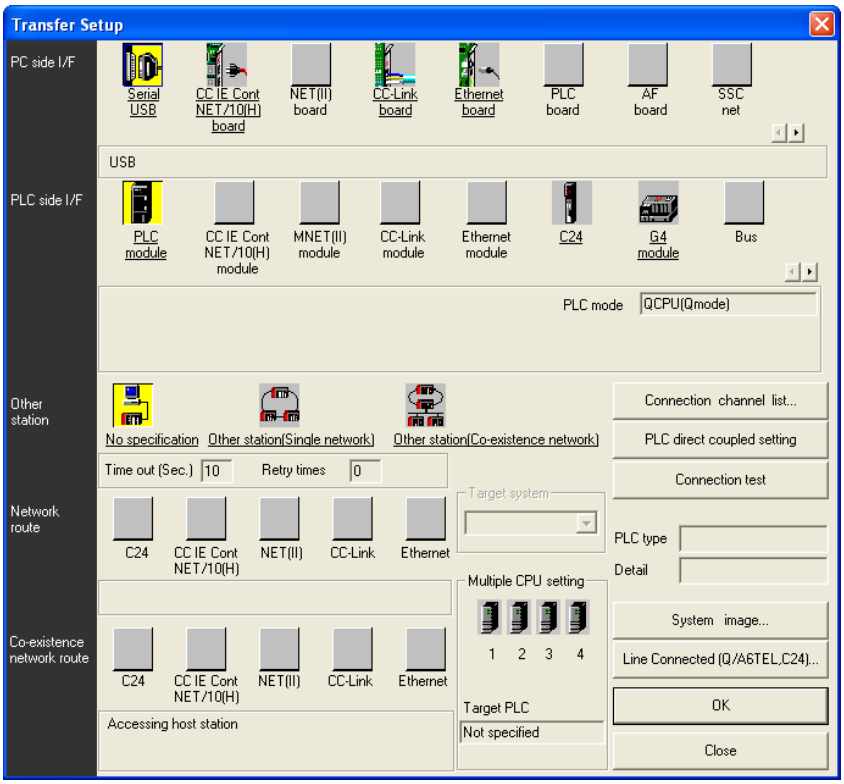


Figure A.7: *Transfer setup ports for the communication between GX IEC Developer and the PLC*

are same as the Figure A.8. Also make sure that the **PLC-Parameter and Program** is chosen as the **DOWNLOAD object** in the **Transfer Setup** dialogue box.

9. Click OK in the Transfer to PLC dialogue box. Then the PLC program will be downloaded to the PLC CPU. You will be prompted by the message box as shown in Figure A.9.

   Click on the **OK** button. After downloading the built PLC program to the PLC CPU you will be prompted by the message box is shown in Figure A.10. Click on the **OK** button.

   After performing the above mentioned steps for downloading the PLC program to the PLC CPU you should get the following message box (see Figure A.11).

   Now the PLC program is successfully downloaded to the PLC CPU but a small error occurred in the PLC, the **ERR**. LED on the PLC CPU starts blinking red and the PLC is put in the STOP state. This error can be removed by ONE of the following procedures:

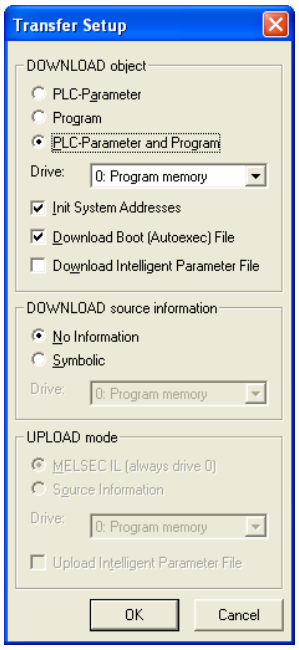


Figure A.8: *Transfer setup project settings*

Figure A.9: *Resetting the PLC message box*

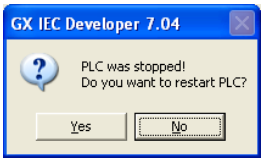


Figure A.10: *Resetting the PLC message box*

(a) **Hardware Procedure**: On the PLC CPU, set the **RESET/L.CLR** switch to RESET to perform a hardware reset and then return the switch to the Neutral position in the middle. This switch is a three state switch which has **RESET, L.CLR** and **Neutral** positions. This switch is normally set to the **Neutral** position. Therefore the **ERR.** LED stops blinking and the PLC CPU returns to the normal mode. Now both of **MODE** and **RUN** LEDs should be green. (See Figure A.12 for

Figure A.11: *Message box which tells that the project is successfully downloaded to the PLC*

more details of the CPU module)

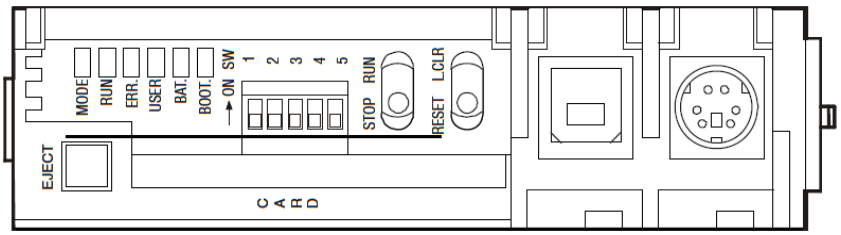


Figure A.12: *12HCPU PLC Module*

(b) **Software Procedure**: Select the **Start / Stop PLC** from the **Online** menu or use the **Alt+S** keyboard shortcut. You will be prompted by the following dialogue box (see Figure A.13).

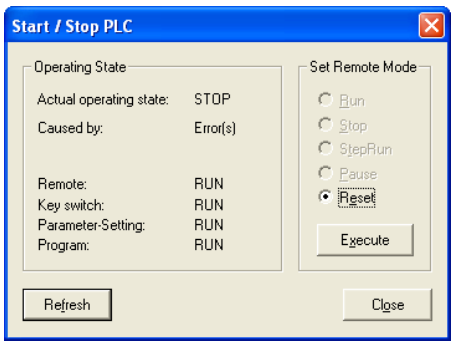


Figure A.13: *Start / Stop PLC dialogue box*

It says that the PLC is in the STOP state due to some errors. Click on the **Execute** button and click the **OK** button on the next confirmation message box. Then the following dialogue box will be shown (see Figure A.14). If the error still remains redo this step again by clicking on the **Execute** button again.

As a matter of fact, resetting the PLC through the hardware procedure is faster than be the software procedure. In addition, one time resetting

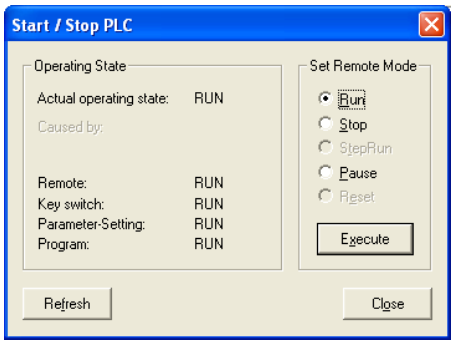


Figure A.14: *Start / Stop PLC dialogue box*

the PLC through software procedure usually does not remove the error and a second try is needed.

Note: It is recommended to RESET the PLC CPU after each simulation run to make sure that all overridden signals restore their initial values.

A.3.2 Monitoring Mode

The PLC program is now ready to run and communicate with DELMIA. This communication is set to be automatic however for monitoring and possibly overriding the signals in the GX IEC Developer program, following steps should be performed:

1. Select **Monitoring Mode** from the **Online** menu or use the **Shift+ESC** keyboard shortcut. Then the status of the desired inputs and outputs in the PLC program can be monitored through the following procedures while **Monitoring Mode** is activated.

2. For instance if one wants to monitor the signals which are in the **S050ModeChoice** POU then **S050ModeChoice [PRG]** should be expanded in the **POU_Pool** tree in the project tree. The Ladder program of the **S050ModeChoice** POU can be opened by double clicking on the **Body [LD]** of the **S050ModeChoice [PRG]**. For this example the place of the S050ModeChoice [PRG] is shown in the project tree in the Figure A.15.

   After opening the **S050ModeChoice** POU in Monitoring Mode, to view the status of the signals in this POU, the monitoring for this specific POU should be activated. In Monitoring Mode when one Ladder or SFC program is opened, Monitoring Mode for that is not activated automatically. To perform the activation of the monitoring for a Ladder or SFC program while it is selected, select **Start Monitoring** from the **Online** menu or use the **Ctrl+F8** keyboard shortcut instead. Then a narrow yellow color border will

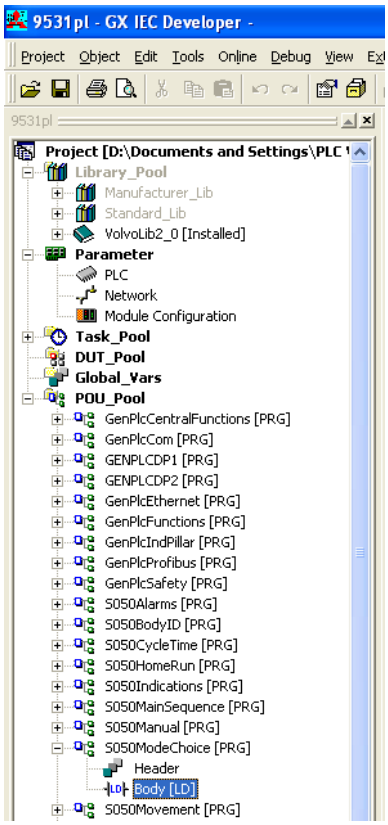


Figure A.15: *Tree view of the PLC program*

appear around of all signals in a Ladder or SFC program which monitoring is started on it (see the Figure A.16).

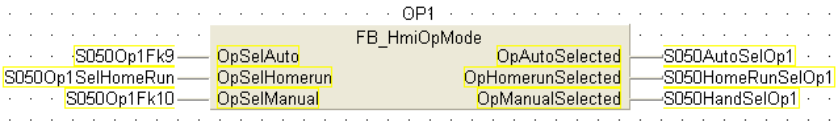


Figure A.16: *Figure shows a function block and its related signals in Monitoring Mode, a yellow outline appears for all signals in Monitoring Mode*

3. In order to change the value of a signal in **Monitoring Mode** there are two methods:

   (a) Double click on a signal's name in a Ladder or SFC program, for which the monitoring is started in its POU (**Ctrl+F8**) or right click on the signal's name and select **Modify Variable Value**. Then if the signals variable type is BOOL, a confirmation message box will appear and

then the signal's value will be changed (see Figure A.17). Then a box filled with yellow appears on the signal's name.

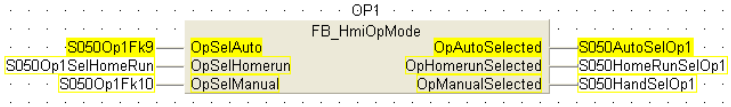


Figure A.17: *Figure shows a function block in Monitoring Mode, for which some of its signals are activated (they are marked with a yellow filled box)*

For the signals with different variable type such as **WORD**, **INT**, etc. a dialogue box will appear for modifying their values (see Figure A.18).

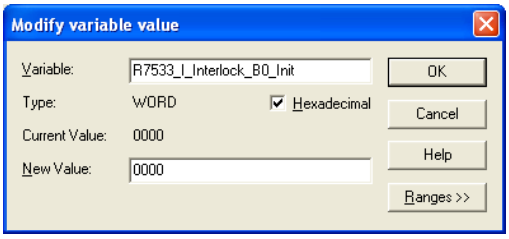
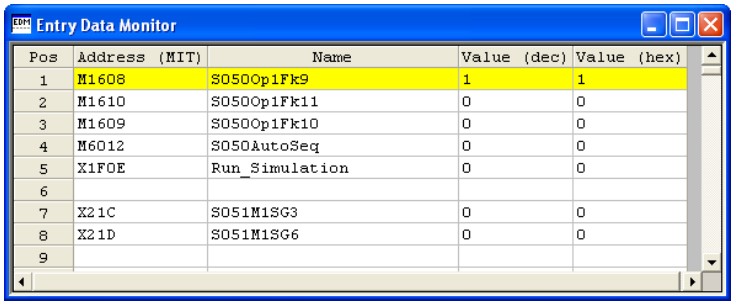


Figure A.18: *A dialogue box which value of a WORD type signal could be overridden through*

(b) Another way to change the value of variables in **Monitoring Mode** is by using the entry data for them. To do this select **Entry Data Monitor** from the **Online** menu or use **Shift+Ctrl+F8** instead. Then the Entry Data Monitor window will appear (see Figure A.19).



Pos	Address (MIT)	Name	Value (dec)	Value (hex)
1	M1608	S050Op1Fk9	1	1
2	M1610	S050Op1Fk11	0	0
3	M1609	S050Op1Fk10	0	0
4	M6012	S050AutoSeq	0	0
5	X1F0E	Run_Simulation	0	0
6				
7	X21C	S051M1SG3	0	0
8	X21D	S051M1SG6	0	0
9				

Figure A.19: *Entry Data Monitor window*

To enter a signal's name to this window one could simply type the signal's name in the Name column and hit **Enter**. Then the address and value of the desired signal will be shown in the respective columns. Another way is to double click on an empty cell in the name column

and then hit the **F2** key. Then a dialogue box will appear and one could find and choose the desired variable name from the object list (see Figure A.20).

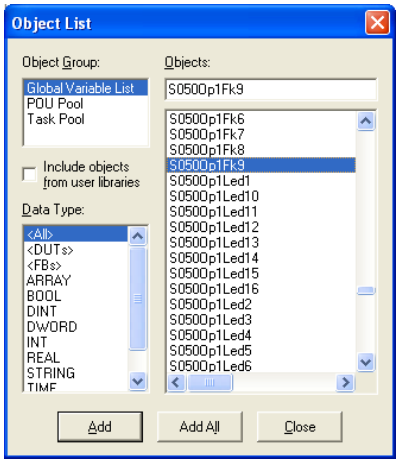


Figure A.20: *Object list of the project; desired signal for insertion in the Entry Data Monitor window can be chosen from the list*

In the **Entry Data Monitor** window the values of signals can be changed by double clicking on a signal's row in the table for the BOOL type variables and double clicking on the value cell of a signal with WORD or INT variable type and typing in the desired value.

Note: changing the value of a signal is possible if and only if that signal is an input (=contact) for the PLC program and not an output (=coil) or outcome of function blocks within the PLC program.

A.3.3 OPC Configuration

1. Run the Beijer Electronics OPC Server program.

2. Select **Open** from the **File** menu to load the configuration file.

3. Load the **Volvo-OPC-Configuration.czc** file which contains the configuration and I/O list for the project. The configuration file contains the information about the PLC and its network IP address which the OPC server is going to communicate with and also the information about the PLC signals which are accessible through OPC communication protocol. After opening the configuration file right click on the **VolvoOPC** in the project tree and select **Property Dialog** (see Figure A.21). Then enter the new IP address and port in the **Configuration** field (see Figure A.22).

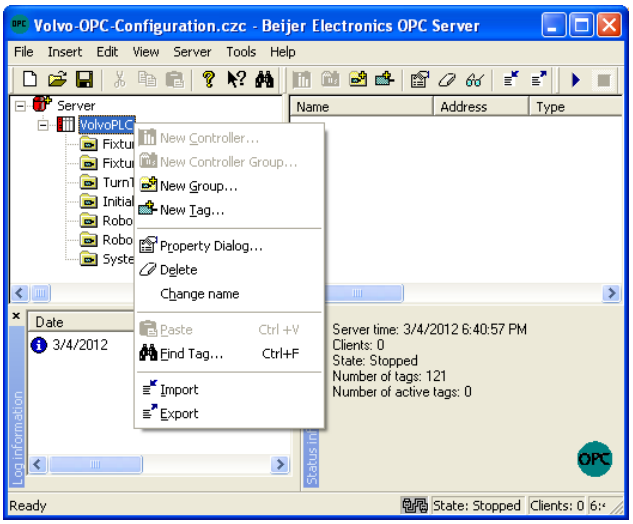


Figure A.21: *Beijer OPC program; how to modify the properties of a controller*

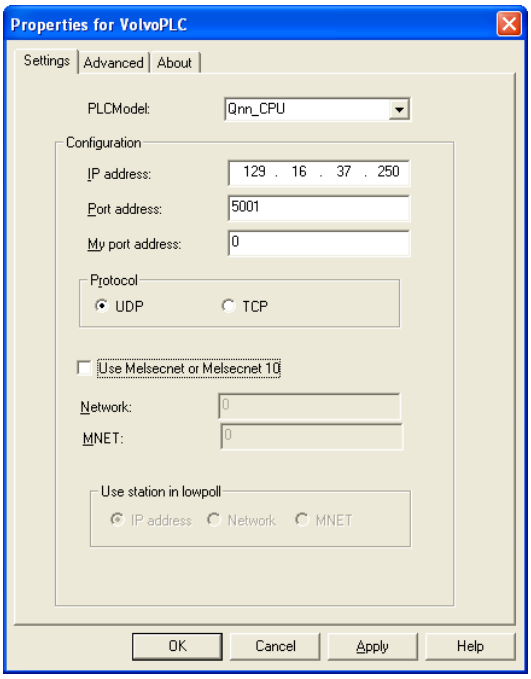


Figure A.22: *Properties dialogue box of the VolvoPLC controller. The IP address of the PLC will be set in this dialogue box.*

4. Select **Run** from the **Server** dropdown menu or press the **F5** key. Then the server is running and the "Controller is connected" message will be shown in the "Log information" window (see Figure A.23).

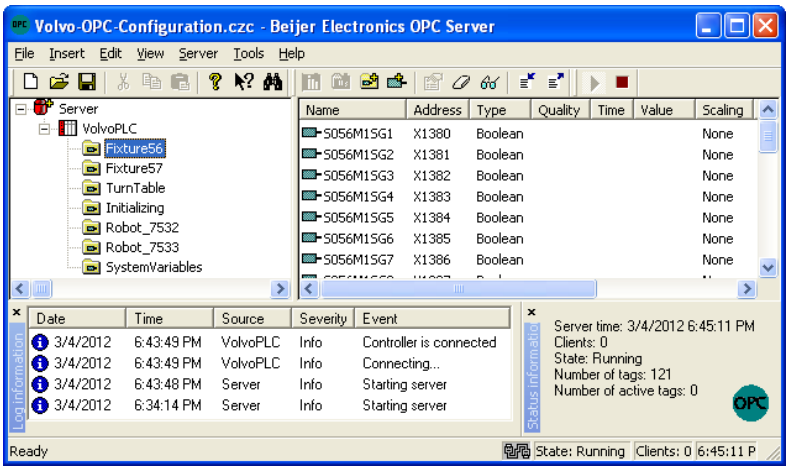


Figure A.23: *Running Beijer OPC server window*

A.4 List of Figures and Tables

List of Figures

List of Tables