

CHALMERS



Virtual Commissioning with Oculus Rift

Bachelor's Thesis

Marcus Björklund
Marcus Engberg
Martin Kastebo
Andreas Lu
Tobias Ågren
Måns Östman

Department of Signals & Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015
Bachelor's Thesis 2015

The Authors grant to Chalmers University of Technology the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet. The Authors warrant that they are the authors to the Work, and warrant that the Work does not contain text, pictures or other material that violates copyright law.

The Authors shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Authors have signed a copyright agreement with a third party regarding the Work, the Authors warrant hereby that they have obtained any necessary permission from this third party to let Chalmers University of Technology store the Work electronically and make it accessible on the Internet.

Virtual Commissioning with Oculus Rift

MARCUS BJÖRKLUND
MARCUS ENGBERG
MARTIN KASTEBO
ANDREAS LU
TOBIAS ÅGREN
MÅNS ÖSTMAN

© Marcus Björklund, Marcus Engberg, Martin Kastebo, Andreas Lu, Tobias Ågren and Måns Östman, June 2015.

Examiner: Ass. Prof. Petter Falkman
Supervisor: Dr. Mohammad Reza Shoaiei

CHALMERS UNIVERSITY OF TECHNOLOGY
Department of Signals and Systems
SE-412 96 Gothenburg
Sweden

Department of Signals and Systems
Gothenburg, Sweden June 2015

Abstract

Virtual Commissioning is a method which both replicates an industrial environment, including hardware and software, into a virtual environment and validates PLC-code. Virtual Commissioning also handles the demands from the market as it improves the time-to-market, planning accuracy, cost constraints and quality. Oculus Rift is a pioneer product among virtual reality goggles where the user gets an improved reality based experience of a virtual world. By introducing Virtual Commissioning with Oculus Rift, we believe a new visual experience for Virtual Commissioning will be achieved.

This thesis studies the opportunity of getting a better perception of reality when performing a Virtual Commissioning on a robot cell. By adding Oculus Rift, we believe, this goal can be achieved. However, a Virtual Commissioning could not be performed and the result of the project was a simulation of the robot cell through Oculus Rift. The impression of the simulation was a realistic experience and further development is of great interest.

Sammanfattning

Virtual Commissioning är en metod där både en industriell miljö, som innehåller hårdvara och mjukvara, replikeras till en virtuell miljö och PLC-kod valideras. Virtual Commissioning möter marknadens efterfrågan genom bättre produktionstid, högre planeringsprecision, mindre kostnader och en bättre kvalitet. Oculus Rift är en pionjär bland virtual reality headsets där användaren erhåller en förbättrad verklighetsupplevelse i den virtuella miljön. Genom att introducera Virtual Commissioning med Oculus Rift, tror vi att man kan få en ny upplevelse med Virtual Commissioning.

Detta projekt undersöker möjligheten att förbättra verklighetsupplevelsen för en Virtual Commissioning av en robotcell. Detta mål anses kunna uppnås genom integrationen av Oculus Rift. En fullständig Virtual Commissioning kunde emellertid ej genomföras, resultatet av projektet var istället en simulering av robotcellen, genom Oculus Rift. Simuleringen gav en verklig uppfattning av den virtuella cellen och framtida utveckling av projektet anses vara av stort intresse.

Acknowledgements

There are a number of people without whom this thesis could not be accomplished and to whom we would like to express our gratitude:

Petter Falkman, for the support and guidance early in the project.

Mohammad Reeza Shoaeei, for the support during the whole project.

Martin Dahl, for the support with the Test Cell and equipment.

Kristofer Bengtsson, for the support with PLC and OPC-server.

Mikael Öhman, for the support with the compiling of the Tcl-C++ wrapper.

Group 15, Gothenburg 2015-06-07

Contents

Glossary	1
1 Introduction	2
1.1 Background	3
1.2 Aims	3
1.3 Methodology	3
1.4 Project Overview	4
2 Implementation	5
2.1 Process Simulate	5
2.2 PLC	9
2.2.1 Project Study	10
2.2.2 Hardware	11
2.2.3 Communication	13
2.3 Oculus Rift	14
2.3.1 Rendering Graphics to Oculus Rift	15
2.3.2 Orientation Control	18
2.4 Integration	26

3 Discussion	27
3.1 Future work	28
4 Conclusion	31
Bibliography	34
Appendices	35
A Test Cell	36
B Oculus Rift - User and Development Guide	103
C Inventory of Safety Equipment	109
D Tutorial: Setting Up a Project in TIA Portal	111
E Tutorial: Setting Up an OPC Server	125
F Tutorial: OPC Tunneling With Matrikon OPC	133
G Tutorial: Validating an OPC Server with OPC Scout	142
H Tutorial: Writing Ladder Code in TIA Portal	147

Glossary

ABB JOKAB is a company specializing in safety equipment.

API is an acronym for Application Programming Interface which is a particular set of rules and specifications that serves as an interface between different software programs and facilitates their interaction.

CAD is an acronym for Computer-Aided Design which is a computer technology that designs a product and documents the design's process.

Event-based is a process which sequence is controlled by signals.

Head Tracking is a method of monitoring the movements of the user's head and translating them into computer input.

OPC Server is a software interface which is used in Windows to interpret PLC protocol, distributed on a server.

PLC is a digital computer used for automation of typically industrial electromechanical processes.

Ramp-up is a term to describe an increase in firm production ahead of anticipated increases in product demand.

Sequence-based is a process which sequence is controlled by operations.

TIA Portal is a software used to configure a PLC and its belonging hardware.

1

Introduction

The demands of better time-to-market, planning accuracy, cost constraints and overall quality of a product are challenges which Virtual Commissioning can overcome [1]. Virtual Commissioning replicates the industrial production environment into a virtual environment, where the PLC code can be validated. By confirming the functionality of the PLC code, hidden errors can be discovered in an earlier stage of the process which results in an increased ramp-up phase and a cost reduction [1] [2]. Virtual Commissioning is useful when implementing new hardware to the production processes since it allows analysis and verifications of the virtual representation on an early stage of development.

The merging of virtual reality and Virtual Commissioning, is an attempt at achieving a more realistic experience of the physical environment. Integrating Oculus rift with Virtual Commissioning gives the opportunity to study the virtual representation from a first person perspective. This perspective introduces chance at gaining new insights into problems which might have been overlooked otherwise.

Oculus Rift is the latest in virtual reality goggles targeting the consumer market which will allow developers to create immersive games and new experiences [5]. Oculus Rift uses sensors in order to achieve 360 degree head tracking. This enables a more realistic perception, since the user can explore the virtual environment through head movements.

1.1 Background

One of the pioneers behind the development of Virtual Commissioning was General Motors, during the 90's [3]. Since then the technology has spread and a new market has appeared where companies develop software for the sole purpose of Virtual Commissioning. One of them is Siemens PLM Software, founder of the software suite Siemens Tecnomatix. Process Design and Process Simulate are part of this suite and was used during this project. Other companies that develops advanced platforms for Virtual Commissioning are ISG Virtuos, Emulate3D, Xcelgo and Visual Components.

Previous studies have been made regarding problems that can occur with the OPC interface [4]. A Study about the implementation of Virtual Commissioning [5] has also been researched, the project herein focuses on implementing a Virtual Commissioning and shares a similar setup to that of this project. These projects have focused on the approach and process of Virtual Commissioning. They will be used as a basis to strengthen the understanding of, and how to conduct, a Virtual Commissioning.

1.2 Aims

The goal of the project is to conduct a Virtual Commissioning on a robot cell in which Oculus Rift is used to view the virtual environment. It is of great interest to evaluate if adding Oculus Rift to a Virtual Commissioning creates a more realistic experience, bringing the user closer to the physical environment.

1.3 Methodology

To reach the end goal of performing a Virtual Commissioning of a robot cell with Oculus Rift, the main task was divided into three subtasks. These will be referred to as; Process Simulate, PLC and Oculus Rift. An overview of the project can be seen in Figure 1.1.

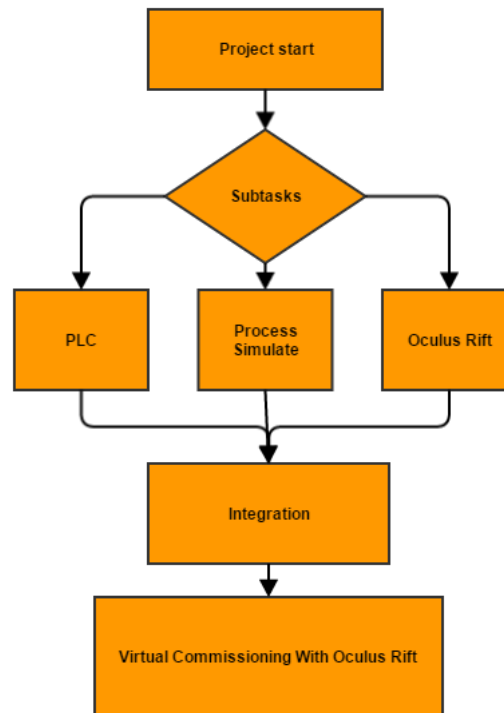


Figure 1.1: Flowchart of the projects workflow.

Each of these three subtasks have different purposes during the project in order to reach the end result. The Process Simulate subtask has the goals to both graphically build a robot cell and also defining its operations. The end result should be a robot cell which is able to be controlled by a physical PLC. The PLC subtask consists of establishing a communication link between the PLC and the virtual environment as well as controlling it. The subtask also includes creating a physical safety environment to be represented as a virtual environment. The Oculus Rift subtask covers the interface between Oculus Rift and Process Simulate and the establishment of an orientation control in the virtual environment.

1.4 Project Overview

The second chapter, consists of methodology and results of each subtask combined with the result of the project. Chapter three, contains discussions and future work of the project. The last chapter contains of conclusions regarding the project.

2

Implementation

The following chapter will describe and present the methods and results for each subtask. In Section 2.1, we describe the subtask Process Simulate. In Section 2.2, we describe the subtask PLC. Finally, in Section 2.3, we describe the subtask Oculus Rift.

2.1 Process Simulate

Siemens Tecnomatix is a software suite to handle part planning, assembly planning, automation and robotics planning, plant design and optimization [6]. Process Design and Process Simulate are software included in Tecnomatix.

Process Design is a tool used to add the required items to a project and to handle how they are associated. This includes the engineering libraries which hold the CAD data and other object within the project.

Process Simulate is a tool used to create the operation which includes both placement of the objects into a 3D environment and defining the tasks performed by the objects. The simulation part can be performed in two ways, either sequence-based or event-based. If the simulation is sequence-based, the tasks defined will run one after another in a predetermined order. It can be used to verify placement through checking for reachability and detecting collisions. Event-based simulations also uses a predetermined order for the tasks, but will only transition from one task to the next if the conditions are met. For instance, the process will not start

unless the start button has been pushed.

Case Study: Test Cell

The SPEAR project was used as a base for studying how to create a robot cell with Tecnomatix[7]. Their methodology is a step by step instruction which covers all the necessary features for creating event based simulation that include safety equipment. It does not include any information on how to replace the virtual PLC within Process Simulate with a real one nor is it prepared for use with Oculus Rift. The purpose of this case study will be to create an event-based simulation of a robot cell that can be adapted to work with a physical PLC along with adding any features required for the integration with Oculus Rift. The robot cell created for this purpose was referred to within the project as the Test Cell.

The contribution to this robot cell simulation model was an inclusion of a human object into the robot cell. This was an essential part due to the integration between Process Simulate and Oculus Rift. By adding the human object into the robot cell, it was possible to control the object and use its in-built function of getting a first person-perspective of the cell.

The Test Cell performs a welding operation on two parts. For this it utilises one fixture, one robot for spot welding and one robot to pick and place the welded parts into a container, see Figure 2.1. The Test Cell also includes several safety features such as two safety gates, one safety mat and one emergency button. The cell operates according to the following sequence:

1. The operator places two different parts onto a fixture
2. The parts are secured in the fixture by clamps
3. A welding robot performs a spot-weld operation, joining the two parts
4. The clamps are opened
5. The gripper robot removes the assembly from the fixture and places it in a container for collection, completing one cycle

By following the instructions provided by the SPEAR project[7], the creation of the Test Cell was divided into three parts. The first step was to use Process Design to allocate all the necessary resources. This was done by creating four folders called Libraries, Product, Plant and Studies. All the provided CAD data was imported to the Libraries folder. The Product folder held the two objects that represent the

parts that would be welded together. The Plants folder contained the objects for the cell and its components along with the objects for the processes defined later. The last folder, Studies, was used to associate objects with the specific simulation created by Process Simulate.

The second part was to use Process Simulate to create a sequence-based representation of the robot cell, which mean its flow is controlled by its sequence of operations. This is commonly the first step of implementation. First all the essential objects to the process were added to the 3D-representation of the robot cell. The process was then defined by using the following operations:

- Object flow operation, handles how the parts are placed into the fixture
- Device control operation, handles the opening and closing of the clamps
- Weld operation, handles the spot-welding robot
- Gripper operation, which handle the gripper robot for removing the finished part

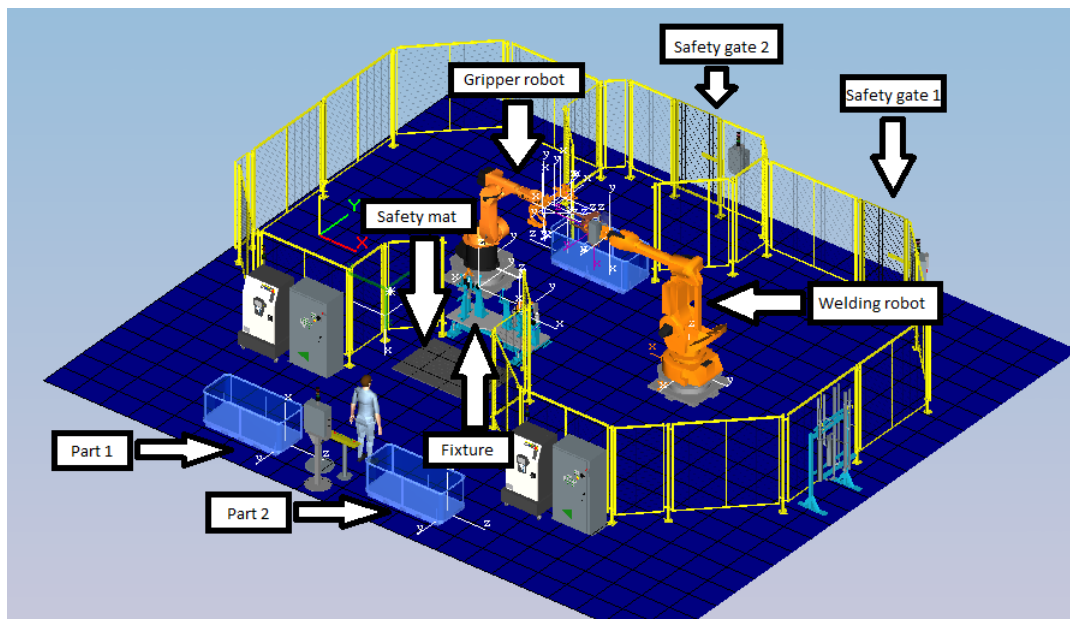


Figure 2.1: The Test Cell.

The last step for the second part was to add the cosmetic feature to give the cell a more realistic look. This includes fences around the cell, gates, containers, PLC's and robot controllers.

The final part was to make the robot cell work as event-based, which means it is controlled by signals. By being controlled by signals, it is possible to connect a PLC to handle the robot cell. To operate a event-based robot cell in Process Simulate, there is a mode called Line Simulation. In Line Simulation it was possible to add signals essential for keeping track of the process. These signals oversee the robot position, status of the clamps and if both parts are present in the fixture. Signals was also generated to control the operations with logic.

The purpose of the result from this subtask is to be integrated with the physical PLC and Oculus Rift. The first goal reached was the sequence-based Test Cell. The graphical result of the Test Cell is depicted in Figure 2.1. Observations of the desired behavior verified the implementation of the sequence along with the associated operations. The desired behavior of the cell was first to push a simulated start button. Secondly, the parts are loaded into the fixture in the cell, where it is locked by the clamps. Then the welding robot perform its welding operation and when finished, it goes to its starting position. The clamps are released and the gripper robot grips the welded part and puts into the container. This is the sequence in which one cycle is performed. The operations implemented to perform this cycle was the following:

1. Start_Button
2. Clamps_Open
3. Part_2_to_Fixture
4. Part_1_to_Fixture
5. Clamps_Close
6. Weld_op
7. Clamps_Open_2
8. Pick_And_Place

The next goal in this subtask was to make the robot cell event-based. By using the sequence-based model of the robot cell, it was possible to continue in Line Simulation mode to make it event-based. In Process Simulate, there is a feature which auto generates the signals based on the implemented operations. Compared to the sequence-based version, it was necessary to implement logic for the transitions

between the operations. Therefore, with help of the signals, a determined sequence could be implemented which can be seen in Figure 2.2. The logic between each transition can be seen in Appendix A.

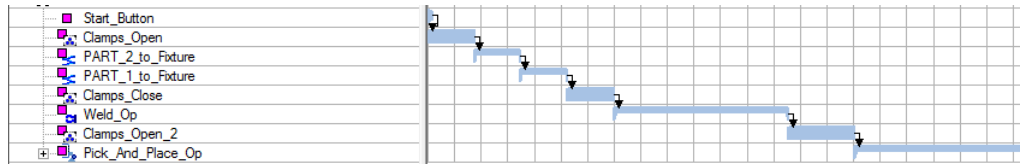


Figure 2.2: The implemented sequence.

With a full functional event-based robot cell, it was possible to implement the safety features. As seen in Figure 2.1, apart from the emergency stop button which not is visible in the picture, the following safety parts are integrated into the robot cell:

- Safety gate 1
- Safety gate 2
- Safety mat
- Emergency stop button

To make these parts integrated to work as safety features, an additional module with corresponding signals was created. In the safety module, logic was created to handle the safety issues in the cell if it is triggered by the safety signals. If triggered, the robots are paused until the signal is reset. When the safety signal is reset, the robots continues the cycle. This can all be seen through a first person-perspective because the importation of the human object into the robot cell. To see the logic for the safety features, see Appendix A.

2.2 PLC

It was estimated that a substantial amount of information had to be gathered in order to control the Test Cell. A study was therefore conducted to develop an understanding of the project as a whole, in order to ease the planning process. The plan was made based on the methods used in The Value Model [8]. This to ensure only essential steps were taken in order to achieve the goals and to minimize the chances of project dead ends. The plan was created through brainstorming

meetings and with the help of feedback provided by the project supervisor as an outside reference.

After the study of the project the process was divided into two parts, hardware and software. This enabled a more focused and efficient work routine, as the process to integrate the PLC with the Test Cell was complex and required in-depth studies.

2.2.1 Project Study

PLC

A study of the PLC system in general was conducted in order to get a basic understanding of how the system functioned. As the PLC is central to the project, knowledge about it and its environment was deemed essential. In order to achieve this, the book Programmable Logic Controllers [9] was used.

Additional Hardware

With basic knowledge of the PLC and its features, provided by the PLC study, an inventory list of the provided ABB JOKAB safety equipment was made. The equipment was categorized in accordance to its relevance concerning the project. This in order to ease the process of finding useful equipment to represent the physical safety environment. The complete inventory list can be found in Appendix C.

Ladder Logic

In this project, ladder logic is used for PLC programming [10]. To this end, a basic understanding of the structure and advantages of the programming language was researched, followed by dedicating time to learn the different function-blocks. The fundamental understanding of the PLC and the ABB JOKAB safety equipment eased the learning process.

Software Interface

Since the PLC subtask was based on a Siemens PLC, TIA portal, also developed by Siemens, was used to manage its configuration. TIA portal was used as an interface between the PLC and the laptop. The instructions to run the Test Cell would be defined in the software, see Appendix G, as well as respond to the signals sent through the PLC from the ABB JOKAB equipment. To communicate the information, an OPC server was established. The server interprets the signals sent from the PLC and a client in Process Simulate in turn interprets the signals and makes the Test Cell respond accordingly. Step by step guides of the work process can be found in Appendix D and E.

2.2.2 Hardware

Siemens PLC and Components

The PLC used in this project is a Siemens SIMATIC 1517F-3 PN/DP. The PLC consisted of several modules interconnected on a rail, see Figure 2.3. Each component was reviewed to get an understanding of how to connect the different safety equipment to the PLC [11].

In addition to studying the components, the PLC interface was studied in order to determine how error messages and different states of the PLC was shown on the display.

Finally, a button was connected to the PLC in order to validate all the gathered information about the system. The button sends a signal when pressed and receive one to light up. A simple ladder program was written to light up the button as it was pressed.

ABB JOKAB Safety Equipment

ABB JOKAB develops safety equipment for industrial production lines. In order to mimic the security setup in the virtual Test Cell, equipment provided by ABB JOKAB was used. The equipment was plugged into the output module and the input module, seen in Figure 2.3, according to the ABB JOKAB manual [12].

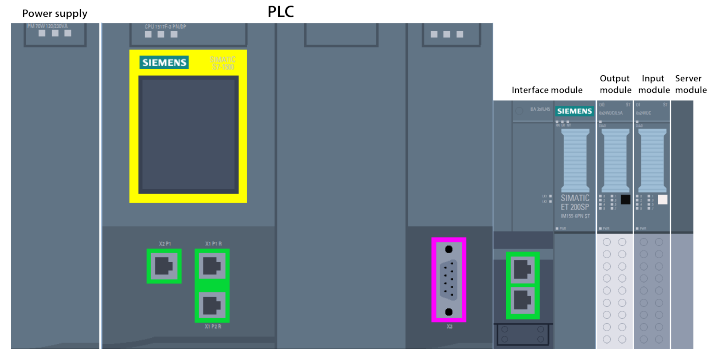


Figure 2.3: Setup of the PLC

The security equipment used was a Focus light curtain FT4-35-300 and a Smile Tina 11 EA emergency stop. These would represent the safety mat as well as the emergency stop in the Test Cell. A turning knob was also used to represent the start button. By programming simple ladder sequences the functionality of the equipment was confirmed. For additional information on ladder code in TIA portal, see Appendix G.

The final setup of the PLC hardware part of the PLC subtask can be seen in Figure 2.4.

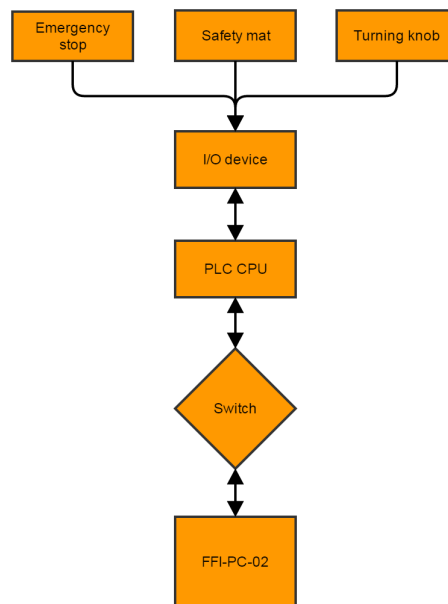


Figure 2.4: The final setup decided upon for the physical safety environment, where FFI-PC-02 is the laptop

2.2.3 Communication

An OPC server was established in the virtual environment of TIA portal, see Appendices D and E.

In order to check the functionality of the OPC server, a client was established through OPC scout, see Appendix H. By using a client on the same system as the server, a range of connectivity issues could be avoided. The server's functionality could be confirmed before the connection between the computers was established.

The link between the computers became the next concern. Different communication alternatives were researched. The concept of this study was to evaluate which connection that proved most stable transferring data between the computers. As the local network was the most accessible alternative it was assessed first. Second an internal VPN-server and finally a local switch, which was decided upon as the best solution.

In order to make the OPC server detectable on the PC station, in which the Test Cell was located, the information on the server was tunnelled between the computers using MatrikonOPC Tunneler, see Appendix F. This allowed the server to be simulated in real time on both computers in order for the client in Process Simulate to read the variables active on the OPC server. The simple ladder sequence displayed in Figure 2.5 was used to test the connection. Process Simulate recognized the variables present on the server, the Test Cell however, did not respond to the signals sent through the OPC server.

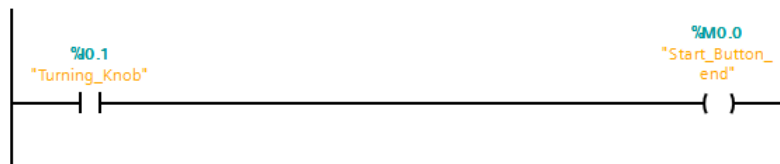


Figure 2.5: Connection test ladder code

The result of the PLC subtask was a communication setup in TIA portal of the PLC and I/O, in which signals was translated to an OPC server and tunnelled using MatrikonOPC Tunneller between two computers. However, since Process Simulate did not respond to the ladder code written in TIA portal, the project was unable to control the Virtual environment of the Test Cell. The final communication configuration can be seen in the Figure 2.6.

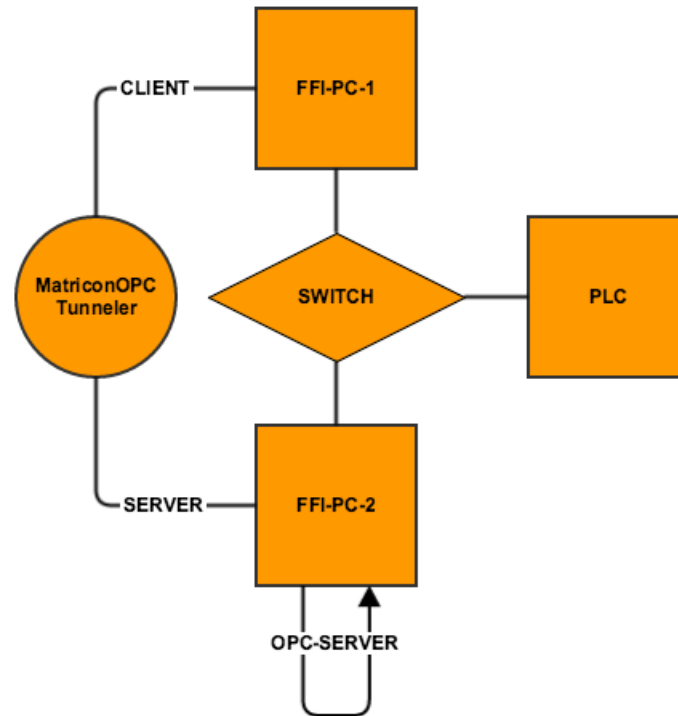


Figure 2.6: Flowchart of the integration between Process Simulate and the physical PLC, where FFI-PC-01 is the PC and FFI-PC-02 is the laptop.

2.3 Oculus Rift

This section covers the implementation of the rendering of graphics from Process Simulate to Oculus Rift and the establishment of an orientation control in Process Simulate.

Using Oculus Rift with Virtual Commissioning is a new approach. An important step was to study the Oculus Rift hardware and software. There are several documents provided by the developers website that possess the necessary information about the hardware and the software [13].

2.3.1 Rendering Graphics to Oculus Rift

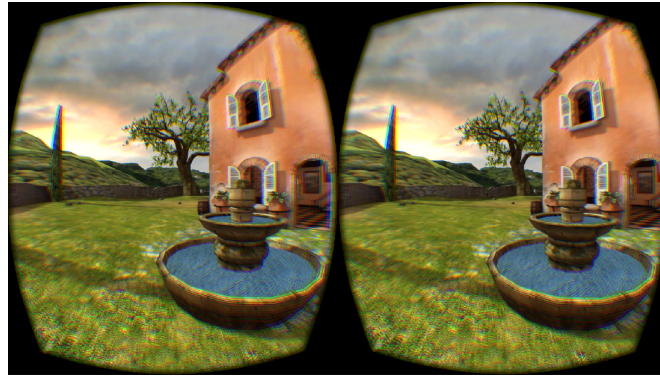


Figure 2.7: Printscreen from OculusWorldDemo stereoscopic rendering.

The first task was to create a first person perspective seen through the Oculus Rift. When using the goggles, each eye has a separate screen to create a stereoscopic view which introduces an illusion of depth as seen in Figure 2.7. Lenses are used to magnify each screen to yield a wider field of view, but they also distort the view significantly.

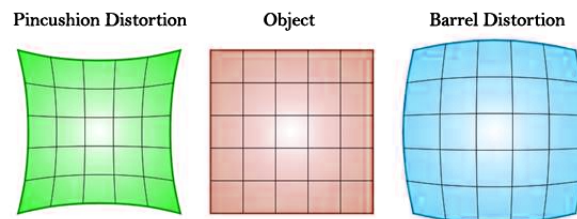


Figure 2.8: Distortion types.

If displaying the original object in Oculus Rift, the user may observe it with pincushion distortion as seen in Figure 2.8. To counteract the pincushion distortion, the software must apply an equal and opposite barrel distortion to cancel each other out. The stereoscopic view and the wide field of view are critical parts of achieving an immersive experience [14].

Setup in Process Simulate

In Process Simulate it is possible to include a human object and change the view to a first person perspective. This will open up two additional windows which represents the left eye and the right eye.

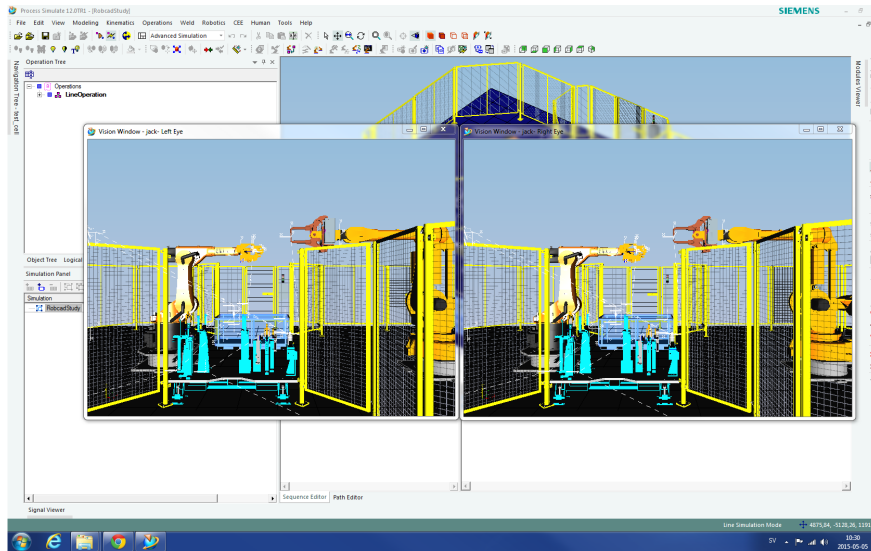


Figure 2.9: Printsreen of the left and right eye output windows in Process Simulate.

Figure 2.9 was the result when the left and the right eye view output windows were generated in Process Simulate. The output windows can not be rendered directly to the Oculus Rift because there is no built-in functionality in Process Simulate. However, Process Simulate lacks the support to develop custom modules. In order to render the graphical view, additional external applications was used.

Rendering Graphics from Process Simulate

In order to render graphics from Process Simulate, a custom application was developed that merges the two windows from Process Simulate into a stereoscopic full-screen image.

An executable Windows application written in the scripting language AutoHotKey (AHK) was chosen. In AHK-libraries, there are already developed API functions that directly can be used to solve parts of the problem. In order to move, remove borders, change dimension and orientation of a window as well as hiding the Windows taskbar and start button, only five short AHK commands was used. The AHK functions are listed in Code 2.1 [15][16][17][18].

Code 2.1: AHK functions used to generate stereoscopic split-screen view.

```
WinGet, OutputVar , Cmd, WinTitle  
WinSet, Attribute, Value , WinTitle  
WinMove, WinTitle, WinText, X, Y , Width, Height  
WinHide ahk_class Shell_TrayWnd ;hide taskbar  
WinHide Start ahk_class Button ;hide start button
```

Figure 2.10 is the result when the stereoscopic full screen view was generated by the custom application. Additional features were added to the application that were hiding the Windows taskbar and start button in order to improve the immersive experience.

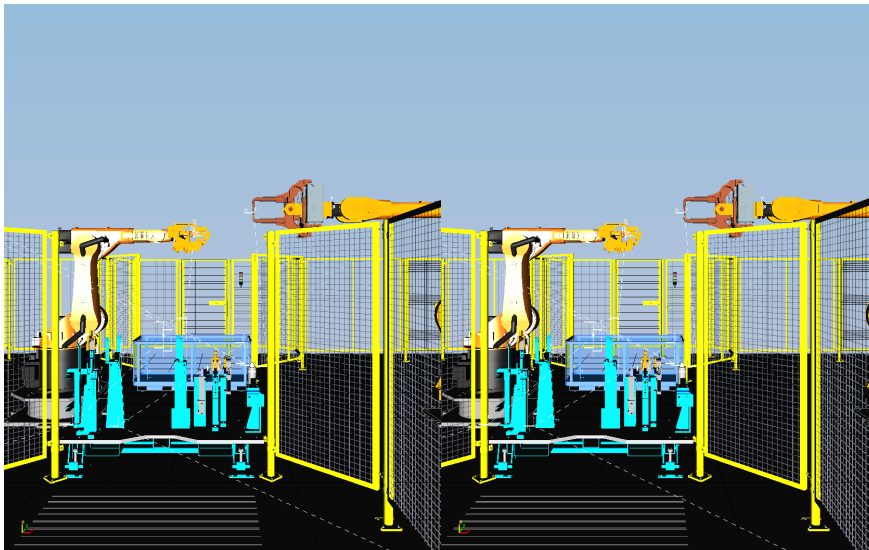


Figure 2.10: Printscreen of the merged stereoscopic full-screen view generated by the custom application.

The stereoscopic split-screen that was generated by the custom application was missing distortion corrections and could therefore not properly be viewed in Oculus Rift. To counterbalance the stereoscopic view, the software Virtual Desktop was used. The software has one setting where distortion corrections can be used on side-by-side content and directly render the view to Oculus Rift. The result of rendering the images in Virtual Desktop can be seen in Figure 2.11.

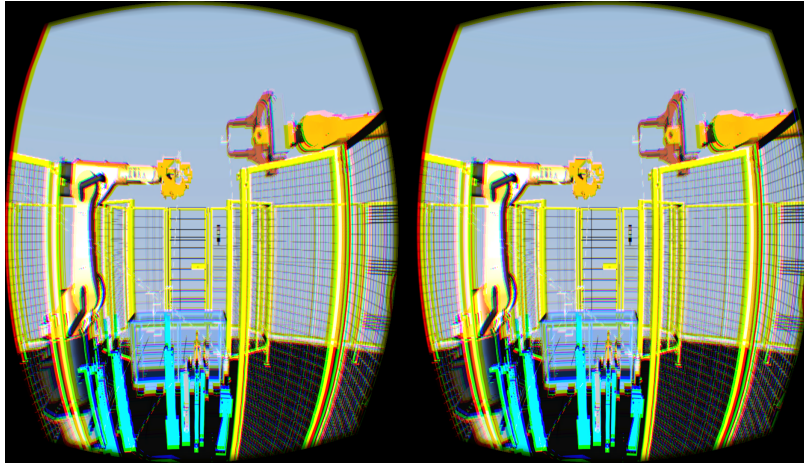


Figure 2.11: The stereoscopic full-screen view with distortion corrections rendered by Virtual Desktop.

2.3.2 Orientation Control

The second task was to control the human object in Process Simulate with help of an input device. One important part was to be able to control the human object during the simulation, otherwise it would be a static view.

Setup Jack Collaboration

There is no way to control the human object directly during simulation in Process Simulate. It is possible to predefine the coordinates for the human object and a movement path for it, but there is no option to give the human object new input data during the simulation.

To solve this issue, an add-on software, Jack 8.2, with a feature called Jack Collaboration was used. This feature connects the human object in Jack 8.2 with the human object in Process Simulate by setting one software as the server and the other as the client. The human object in the client was controlled from the software set as the server which gave the opportunity to control the orientation of the human object in Process Simulate from Jack 8.2.

Controlling Orientation in Jack 8.2

A Jack 8.2 Tool Command Language (Tcl) module was developed in Jack 8.2 in order to control the orientation. The Jack 8.2 module system is based on Tcl packages. This functionality is extended with hooks for menu creation and other types of module initialization tasks [19].

In Jack 8.2, it is possible to manually change the orientation of a human by using the functions in the menu bar. Following Tcl commands can then be viewed in the log viewer seen in Code 2.2.

Code 2.2: Tcl commands from the log viewer.

```
set jcHuman_4 [jcScene_findHuman [jcGlobal_getScene] human]
set jMatrix_5 [new_jMatrix]
jMatrix_setList $jMatrix_5 {list}
jcHuman_setLocation $jcHuman_4 $jMatrix_5
delete_jMatrix $jMatrix_5
```

After analyzing and experimenting with the Tcl commands, the parameters used can be explained by Table 2.1.

Table 2.1: Explanations of the parameters from the log viewer.

Parameters	Explanation
jcHuman_4	The human object that is controlled.
jMatrix_5	Matrix object that is temporary used.
list	A list of 16 elements.

$$\text{list} = \{x \ y \ z \ 0 \ R_{11} \ R_{12} \ R_{13} \ 0 \ R_{21} \ R_{22} \ R_{23} \ 0 \ R_{31} \ R_{32} \ R_{33} \ 1\}$$

The x, y, and z values in the list are the translation in three perpendicular axes and the R values represents the rotation about three perpendicular axes. In order to easily control the rotational orientation of the human object in Jack 8.2, it was necessary to set up a function that transforms Euler angles into a rotation matrix. Euler angles are three angles that can be used to describe any rotation.

A general rotation matrix has the form that can be seen in Equation (2.1).

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \quad (2.1)$$

This matrix is a sequence of three rotations, one about each principle axis. Since matrix multiplication does not commute, the order of the axes that one rotates about will affect the result. Below, the rotation will first be about the x-axis, then the y-axis and finally the z-axis. This sequence of rotations can be represented as the matrix product in Equation (2.2) [20].

$$R = R_z(\phi)R_y(\theta)R_x(\psi) = \tag{2.2}$$

$$= \begin{bmatrix} \cos(\theta)\cos(\phi) & \sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi) & \sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi) \\ \cos(\theta)\sin(\phi) & \sin(\psi)\sin(\theta)\sin(\phi) + \cos(\psi)\cos(\phi) & \cos(\psi)\sin(\theta)\sin(\phi) - \sin(\psi)\cos(\phi) \\ -\sin(\theta) & \sin(\psi)\cos(\theta) & \cos(\psi)\cos(\theta) \end{bmatrix}$$

Input Devices

To control the human object with a keyboard or other input device, Jack 8.2 has to perceive signals from the device. There are two potential approaches that can be used in order to set up an interface between the input device and Jack.

1. Use the built in Tcl function **bind**.
2. Develop more advanced functions by extending the Tcl language.

The second approach was not chosen to be followed because it is much more complex than binding input signals.

The function **bind** listens to keystrokes from a keyboard directly in the Tcl module. The binding approach is about setting up an interface, between a keyboard or another input device and Jack, by binding signals to certain actions. To bind a keyboard key to a certain event, **bind** can be used by the Tcl example code seen in Code 2.3. The parameters are explained in Table 2.2 [21].

Code 2.3: Tcl binding function example.

```
bind all <key_name> {
MyFunction
}
```

Table 2.2: Explanation of the binding function parameters

Parameters	Explanation
key_name	name of key to bind
all	bindtag for global bindings
MyFunction	function to call on when key signal is received

By binding a key to call on a custom function, it is possible to remap a specific keyboard key to a certain action. Pressing the keyboard key [W] could for instance correspond to moving the human object a few centimetres forward. There were, however, issues with the key-binding approach that had to be taken into account.

When a key is pressed down, two different time delays makes the binding function poor. The first delay is the initial pause before the repeat starts and the second delay is the delay between each repetition. Both delays had to be eliminated completely in order to get a smooth orientation control. Delays associated with key pressing could almost be eliminated with the AHK-command seen in Code 2.4.

Code 2.4: AutoHotKey function used to eliminate keyboard key delays.

```
SetKeyDelay , -1
```

Using the input device with the **bind** function, the signals from the input device needed to be remapped to keyboard key signals, in order to be recognized by the function. There is a built-in remapping feature in the AHK language that was used to remap both keyboard and input device keys. Remapping a keyboard key can be done by the AHK example code seen in Code 2.5.

Code 2.5: AutoHotKey remapping example.

```
a :: b
```

The example above will make the [A] key to behave like the [B] key, but it will not alter the [B] key itself.

The same command can also be used for input device keys, however, the device key names must first be determined. There was an already developed test script, that was executed to help determine the button numbers and names for an input device. The script was named JoystickTest.ahk and was found at AutoHotKey website [22]. When input device key names were determined, they were used with the built-in remapping feature exactly the same way as for the keyboard keys.

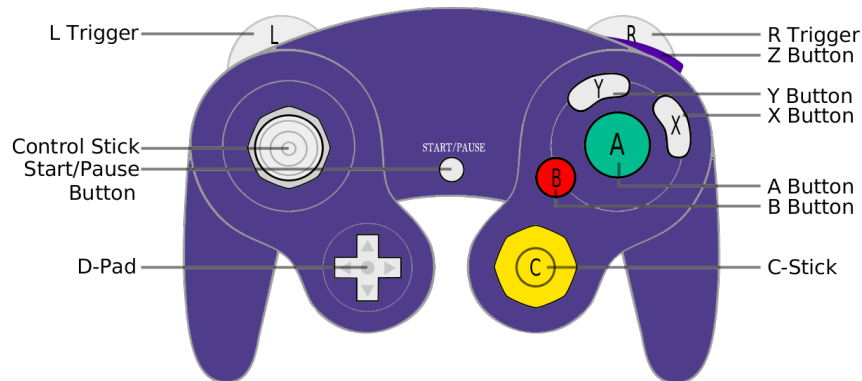


Figure 2.12: Nintendo GameCube controller button names.

A Nintendo GameCube controller was chosen as input device. All corresponding device key names to the button names, seen in Figure 2.12, was determined by running the JoystickTest.ahk script. Code 2.6 is an example where the first numbered button, (X Button), on a Nintendo GameCube controller is remapped to correspond to the [F1] keyboard button.

Code 2.6: AutoHotKey remapping example.

```
Joy1 :: send {F1}
```

In the custom application, keyboard arrow keys and Nintendo GameCube buttons and joysticks were remapped according to the scheme in Table 2.3.

Table 2.3: Remapping scheme in the custom external application.

Nintendo Game-Cube key input	Key output	Keyboard key input	Key output
joyY < 30	Send Up arrow	Up arrow	Send W
joyX < 30	Send Left arrow	Left arrow	Send A
joyY > 70	Send Down arrow	Down arrow	Send S
joyX > 70	Send Right arrow	Right arrow	Send D
joyR < 30	Send Numpad4	Numpad4	Send F
joyR > 70	Send Numpad6	Numpad6	Send G
L-Trigger	Send Numpad8	Numpad8	Send I
R-Trigger	Send Numpad2	Numpad2	Send K
		Up & Left arrow	Send Q
		Up & Right arrow	Send E
		Down & Left arrow	Send Z
		Down & Right arrow	Send C
		Up & Down arrow	do nothing
		Left & Right arrow	do nothing

The custom application made the input keys to behave as the output keys and sends the output to the Oculus Tcl module Jack 8.2. The Nintendo GameCube L/R-trigger, control and c-stick sends keyboard arrow keys and numpad keys which in turn sent keyboard key inputs. The joyX, joyY, joyR key input are the key states of the Nintendo GameCube control and c-stick and is explained in appendix B, section 5.2.

Table 2.4 shows all key bindings that was set up in the Oculus Tcl module in Jack 8.2.

Table 2.4: Binding scheme in Oculus Tcl module.

Key binding	Bound function event	Step size
W	Move forward	5 cm
A	Move left	5 cm
S	Move backward	5 cm
D	Move right	5 cm
Q	Move forward and left	5 cm diagonally
E	Move forward and right	5 cm diagonally
Z	Move backward and left left	5 cm diagonally
C	Move backward and right	5 cm diagonally
F	Rotate left	1°
G	Rotate right	1°
I	Levitate up	5 cm
K	Levitate down	5 cm

Incoming keyboard signals from the custom application trigger function events, via the key binding feature, and reorients the human object in Jack 8.2 according to the step size in the table.

Oculus Rift Head Tracking

In order to use the Oculus Rift head tracking feature for orientation control in a new application, following steps must be performed according to the Oculus developer guide [14]:

1. Initialize LibOVR.
2. Enumerate Oculus devices, create the `ovrHmd` object and configure tracking.
3. Integrate head tracking into the developed application's movement code which involves:
 - (a) Reading data from the Rift sensors through `ovrHmd_GetTrackingState` or `ovrHmd_GetEyePoses`.
 - (b) Applying Rift orientation and position to the camera view while combining it with other application controls.
 - (c) Modifying movement and game play to consider head orientation.

The steps (1), (2) and (3a) can be achieved by applying the theory found on pages 17-23 in the Oculus developer guide [14].

Step (3b) and (3c) can be obtained by writing a Tcl - C++ wrapper so all necessary Oculus routines can be accessible via Tcl.

Tcl has the ability that it can be extended with code written in C++. It also provides an interpreted environment that makes it possible to interactively control and manipulate the underlying C++ application such as calling functions or examine variables. In newer versions of Tcl, this is usually done by compiling the C++ extension module into a shared library (.so-file) or dynamic-link library (.dll-file) that dynamically can be loaded into the Tcl environment [23].

In order to load dynamic-link libraries to initialize new commands, the Tcl command **load** was used seen in Code 2.7 [24].

Code 2.7: Tcl load command example.

```
load fileName.dll
```

There was already a developed project solution for this at Github [25] that was used. The project is a Tcl - C++ wrapper that compiles a dll-file, that can be loaded in Tcl, which makes the application main routines accessible via Tcl. All available routines in this wrapper were not of interest and some parts of the contents could either be removed or slimmed. All rendering routines were removed and routines connected to the Oculus Rift head tracking feature were kept. For detailed instructions of how this project was compiled, see Appendix B, Section 5.3.

When the wrapper is compiled and loaded into the Tcl environment, it would be possible to initialize new Tcl commands. The Tcl command in Code 2.8 initializes Oculus Rift.

Code 2.8: New Tcl command that initializes Oculus Rift and return sensor data.

```
tclovrGetData
```

It returns the current sensor data to Tcl in the format: tx ty tz rx ry rz, where tx, ty and tz is the offset and rx, ry and rz is the rotation of the Oculus Rift goggles. When the offset and rotation data is passed from the Oculus Rift sensors into the Tcl environment, the data can directly be used to update the orientation of the human object by passing it to functions which updates the orientation.

Step (3b) and (3c) was not fully established in this project and the head tracking feature was therefore not integrated in the solution.

The Figure 2.13 is a overview flowchart of the whole interface between the hardware and software.

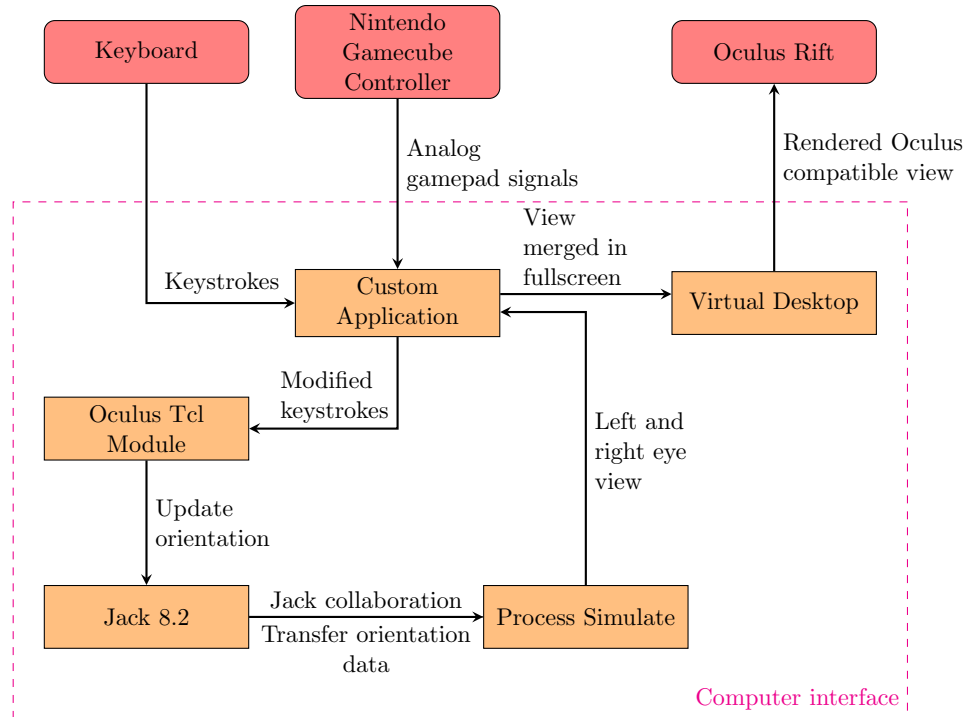


Figure 2.13: Overview flowchart. The red boxes corresponds to hardware and the orange boxes corresponds to software.

2.4 Integration

The final part of the project, as seen in Figure 1.1, was integrating the different subtasks into a solution. Since the communication between the PLC and Process Simulate was not established, a Virtual Commissioning was not performed. The integration part of the project was instead solely the simulation of the Test Cell viewed through Oculus Rift, in which the head tracking feature was excluded.

3

Discussion

The end result did not match the expectation stated at the start of the project. Each sub task encountered issues during the project which changed the outcome. However, we successfully made an event based robot cell and are able to use the Oculus Rift to experience the simulation from a first person perspective. We also designed an orientation control in order to manoeuvre the human object during the simulation and therefore achieve a more realistic experience of the robot cell.

We believe that the methodology from the SPEAR project [7], the step-by-step instruction about building the Test Cell, facilitated the learning process. We first acquired theoretical knowledge about every step and then applied it practically on the Test Cell. However, we encountered some problems. The fixture in the cell had no given coordinates for its placement. We could, after some time, solve the problem and get the approximate coordinates for its placement. The result from the subtask, Process Simulate, was the event-based robot cell with an inclusion of a Human Object. With the inclusion of the Human Object, the simulation could be viewed in a first person-perspective which gave a more realistic perception. This made us even more curious of how it would be experienced with the integration of Oculus Rift.

Establishing a communication path between the PLC and Process Simulate turned out to be a much larger task than anticipated. Therefore not much time was left to developing the physical safety environment. Another reason for delays was caused trying to get the OPC setup to function over wireless and later using a VPN connection. This was found impossible due to security configuration of the wireless network.

Since the setup of the PLC, OPC server and OPC tunnel between the PC and the laptop was successful we believe that the majority of the connection setup is finished. We found it probable that the communication failed due to wrong settings in either Process Simulate or TIA portal.

A few irregularities arose when testing the safety equipment. The light emitting diodes did not work as intended. This was rather trivial to the project, as the main objective was to create a Virtual Commissioning. In a real robot cell, however, such issues do have to be solved, since operators need to quickly determine the state of different equipment. This also makes a good argument to use the ABB JOKAB safety PLC. Using such a system can also allow a far more realistic setup which is of interest, since the PLC code easier can be correctly validated.

The integration of the Oculus Rift head tracking feature was never fully developed for the orientation control purpose. The Tcl - C++ wrapper did not include all necessary project dependencies such as Oculus and Tcl library and source files nor full instructions on how to compile the project. Without full instructions, we had to debug the code by trial and error. In this respect, we had to figure out which project settings and library file versions that was needed to be used for a successful compilation.

When the Tcl - C++ wrapper was compiled, the result was still not as expected. The dynamic-link library could not properly be loaded into Jack 8.2 and no helpful error messages was given while the program was running. Without necessary information, it was hard to debug the problems. Due to the lack of time and programming knowledge, the group decided to not pursue this issue.

3.1 Future work

In order to perform a Virtual Commissioning a thorough investigation of why the communication didn't work as intended needs to be carried out. To do this efficiently a different setup could be considered where the commissioning only uses one computer therefore eliminating the need for an OPC tunnel. When the issues have been resolved the approach of using an OPC tunnel to transfer the PLC data could be implemented once more.

When a Virtual Commissioning has successfully been performed the next step is to start considering how to allow a better user experience. We have noted problems using Oculus Rift while working with the PLC-equipment. Since the Oculus Rift limits the users eyesight to the virtual world, orientation in the physical environment becomes difficult. The user has a hard time interacting with different

HMI's and it would therefore be of great interest to investigate how to properly interact with physical components, while using Oculus Rift.

To improve the experience of using Oculus Rift when viewing a virtual environment implementing head tracking would be of great interest. In order to achieve this, a dynamic link library needs to be established. Furthermore we would strive towards creating a more easy-used, slimmed solution by decreasing the amount of interactions between the software modules. This should be implemented so that it does not effect the result and the user experience in a negative way. Figure 3.1 is a flowchart of a possible optimized solution.

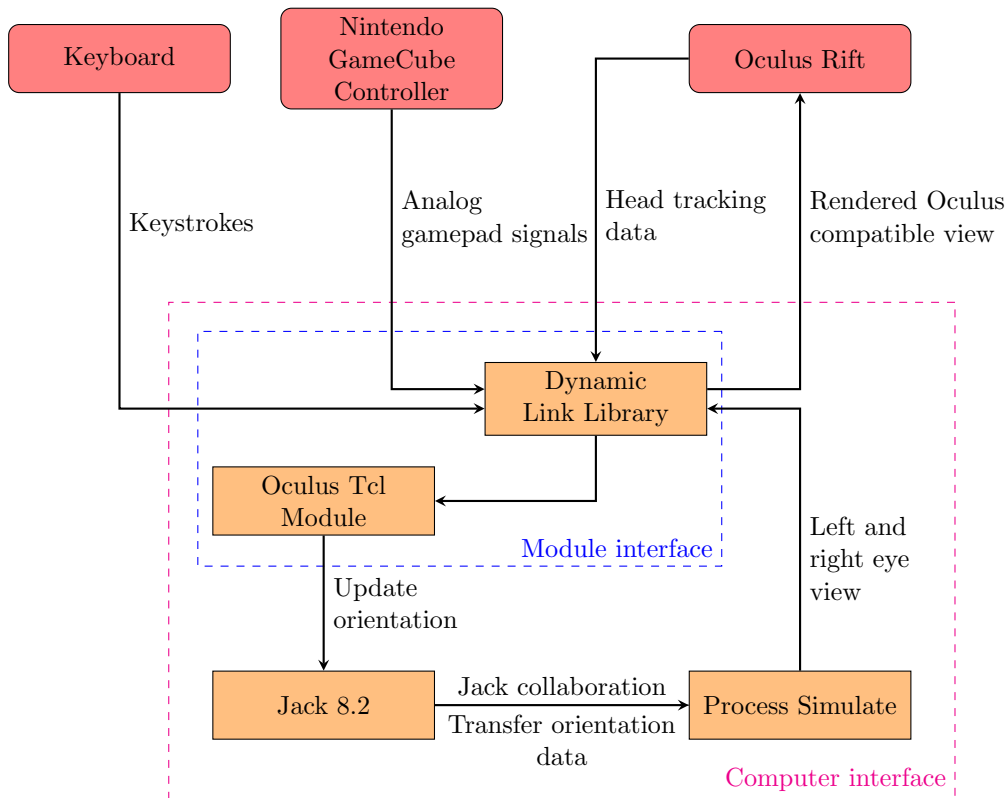


Figure 3.1: Overview flowchart of optimized solution.

The differences to the current solution are fewer software modules and an integrated Oculus Rift head tracking feature. The custom application and Virtual Desktop is replaced by a dynamic-link library that can handle both graphics rendering to Oculus Rift and process signals from the keyboard and Nintendo GameCube controller.

To make the transition between the plane movement directions more smooth, it would be better to read the analog state value of the Nintendo GameCube control

stick, pass and use it in the Tcl module so corresponding movement direction can be applied. It would also be better to separate plane and rotational movement controls by controlling rotation by mouse or Nintendo GameCube C-stick and plane movement by keyboard arrow keys or Nintendo GameCube control stick. This would make the rotational movement control more sufficient since it only is possible to rotate about one axis in the current solution.

We believe that using Oculus Rift with implemented head tracking support together with a well designed physical interface will lead to a better perception of a virtual model. This could give a new perspective when inspecting the behaviour of industrial processes.

4

Conclusion

During the time of this project, we have studied the possibilities of getting a better perception of reality through Virtual Commissioning with a simulation in Oculus Rift. Virtual Commissioning is a method that improves the ramp-up phase, quality, planning accuracy and cost constraints compared to a physical commissioning. In order to determine if combining Virtual Commissioning and the Oculus Rift will give a closer perspective towards reality, the task had to be divided into three parts.

The result was a simulation of the robot cell with Oculus Rift. However, we were not able to integrate the Oculus Rift head tracking which led to a less realistic experience. Despite the lack of head tracking the simulation experience was confiding. Through the simulation, we were able to explore the virtual reality and see the robot cell operate in real time. The experience was more realistic compared to watching the result on a computer monitor, however, not as real as we expected at the start of the project. We believe it depends on the lack of the head tracking feature.

Future work could be to get a PLC integrated with Process Simulate, integrating the Oculus Rift head tracking feature with the orientation control. If these features would be developed into this project, we believe it would give a more determinate solution towards our aim.

Bibliography

- [1] G. Reinhart, G. Wunsch, Economic application of virtual commissioning to mechatronic production systems (Nov. 2007).
URL <http://link.springer.com/article/10.1007/s11740-007-0066-0/fulltext.html>
- [2] S. Makris, G. Michalos, G. Chryssolouris, Virtual Commissioning of an Assembly Cell with Cooperating Robots (2012).
URL <http://www.hindawi.com/journals/ads/2012/428060/>
- [3] L. Jing, Y. Chengyin, G. Fangming, B. Stephan, W. Demet, A. Daniel, Virtual commissioning: Gm.
URL http://worldwide.espacenet.com/publicationDetails/biblio?CC=US&NR=8949480&KC=&FT=E&locale=en_E
- [4] C. Henrik, S. Bo, D. Fredrik, L. Bengt, Methods for reliable simulation-based plc code verification (May 2012).
URL <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6121945>
- [5] H. Ali, S. Oliver, Virtual commissioning of an existing manufacturing cell at volvo car corporation using delmia v6 (May 2012).
URL <http://publications.lib.chalmers.se/records/fulltext/157312.pdf>
- [6] Siemens, Tecnomatix12 (2015).
URL https://www.plm.automation.siemens.com/en_us/products/tecomatix/tecomatix12/index.shtml
- [7] R. Cawley, E. McDougall, J. Pratt, R. Weir, SPEAR.
- [8] P. Lindstedt, J. Burenus, The Value Model - How to Master Product Development and Create Unrivalled Customer Value, Nimba, 2004.

- [9] W. Bolton, Programmable Logic Controllers, Newnes, 2009.
- [10] R. W. Lewis, Programming industrial control systems using IEC 1131-3, The Institution of Electrical Engineers, London, 1998.
- [11] Siemens, Simatic et 200sp, distributed i/o system (Mar. 2015).
URL https://support.industry.siemens.com/dl/files/293/58649293/att_844992/v1/et200sp_system_manual_en-US_en-US.pdf
- [12] Jokab Safety AB, Säkerhetshandboken, Erfarenhet - System - Produkter (2008).
- [13] O. VR, Oculus vr: Pc sdk documentation.
URL <https://developer.oculus.com/documentation/>
- [14] L. Oculus VR, Oculus Developer Guide (Dec. 2014).
URL http://static.oculus.com/sdk-downloads/documents/Oculus_Developer_Guide_0.4.4.pdf
- [15] AutoHotKey, WinGet function description.
URL <https://www.autohotkey.com/docs/commands/WinGet.htm>
- [16] AutoHotKey, WinSet function description.
URL <https://www.autohotkey.com/docs/commands/WinSet.htm>
- [17] AutoHotKey, WinMove function description.
URL <https://www.autohotkey.com/docs/commands/WinMove.htm>
- [18] AutoHotKey, WinHide function description.
URL <https://www.autohotkey.com/docs/commands/WinHide.htm>
- [19] Siemens, Developing Modules in Jack (Jun. 2014).
- [20] G. G. Slabaugh, Computing Euler angles from a rotation matrix (Oct. 2014).
URL <http://staff.city.ac.uk/~sbbh653/publications/euler.pdf>
- [21] A. Goth, Bind function description (Jun. 2014).
URL <http://wiki.tcl.tk/1401>
- [22] AutoHotKey, Joystick Test Script (May 2005).
URL <https://www.autohotkey.com/docs/scripts/JoystickTest.htm>

- [23] D. M. Beazley, Tcl and SWIG as a C/C++ Development Tool (1998).
URL <http://www.swig.org/papers/Tcl98/TclChap.html>
- [24] P. Yorick, Load function description (Dec. 2013).
URL <http://wiki.tcl.tk/1208>
- [25] M. Yzusqui, OculusVr-CmDLL Project (Jan. 2015).
URL <https://github.com/myzb/OculusVr-CmDLL/tree/master/OculusVr-CmDLL>

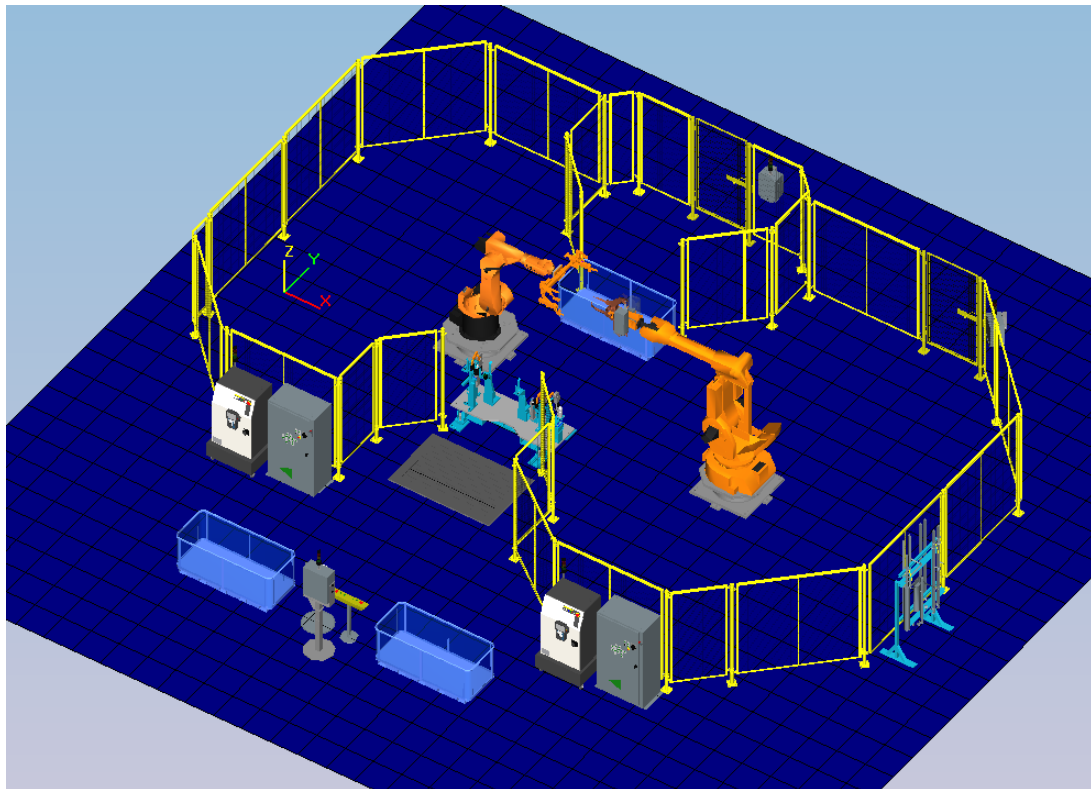
Appendices

A

Test Cell

VIRTUAL FACTORY TUTORIAL

A STEP BY STEP DEVELOPMENT OF A DIGITAL PRODUCTION CELL



CONTENTS

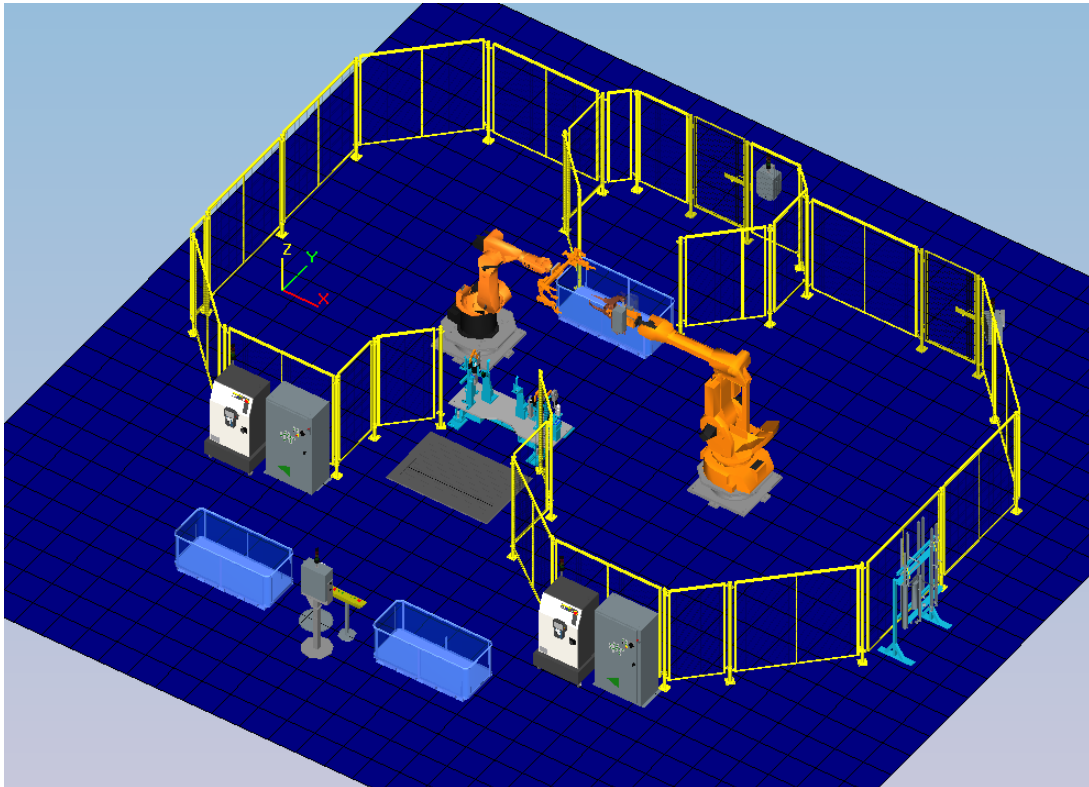
CHAPTER ONE - INTRODUCTION	1
1.0 SOFTWARE OVERVIEW.....	2
1.1 PROCESS DESIGNER.....	2
1.2 PROCESS SIMULATE.....	2
1.3 THE TECNOMATIX SETUP.....	2
2.0 BASIC DEFINITIONS.....	3
2.1 OPENING THE SOFTWARE.....	3
2.2 VIEWERS.....	4
2.3 NODES.....	5
2.4 USING THE MOUSE.....	5
2.5 SYSTEM ROOT FOLDER.....	5
2.6 SAVING YOUR SIMULATION.....	6
2.7 IMPORT/EXPORT OF SIMULATIONS.....	6
3.0 PRE-OPENING TASKS.....	6
4.0 CREATING A PROJECT.....	6
5.0 CREATING THE FOLDER STRUCTURE.....	7
6.0 CREATING ENGINEERING LIBRARIES.....	8
7.0 CREATING THE PROJECT NODES.....	9
8.0 ALLOCATING PARTS AND RESOURCES.....	10
9.0 OPENING IN PROCESS SIMULATE.....	13
10.0 POSITIONING OF RESOURCES.....	14
CHAPTER 2 – CREATING OPERATIONS	17
11.0 CREATING OPERATIONS.....	18
11.1 OBJECT FLOW OPERATIONS.....	18
11.2 DEVICE CONTROL GROUP OPERATIONS.....	19
11.3 WELD OPERATIONS.....	23
11.4 GRIPPER OPERATIONS: PICK AND PLACE.....	29
12.0 PLACEMENT OF PROXIMITY SENSORS.....	30
13.0 ADDING COSMETIC FEATURES.....	31
14.0 REFINEMENT OF PATH LOCATIONS.....	33



14.1 OBJECT FLOW OPERATION.....	33
14.2 WELD OPERATION.....	34
14.3 PICK AND PLACE LOCATIONS.....	35
CHAPTER 3 – CYCLIC EVENT EVALUATOR	37
15. DEFINITION OF MATERIAL FLOW	38
16.0 DEFINITION OF JOINT VALUE SENSORS	39
16.1 ABB_IRB64 ROBOT JOINT VALUE SENSORS	39
16.2 KR30_3 ROBOT JOINT VALUE SENSORS	41
17.0 PROXIMITY SENSORS	42
18.0 SIGNAL GENERATION.....	43
18.1 CREATION OF ROBOT SIGNALS.....	43
18.2 DEVICE SIGNALS	45
18.3 USING THE SIGNAL VIEWER	46
18.4 CREATING THE START CYCLE SIGNAL.....	46
19.0 INTRODUCING LOGIC TO THE SIMULATION.....	47
19.1 TRANSITIONS.....	47
19.2 ROBOT PROGRAMS	51
19.3 MODULES	54
20.0 INTRODUCING SAFETY FEATURES.....	57
20.1 GATES AND SAFETY MAT	57
20.2 EMERGENCY STOP BUTTON.....	59
21.0 PRACTICE ACTIVITY	60

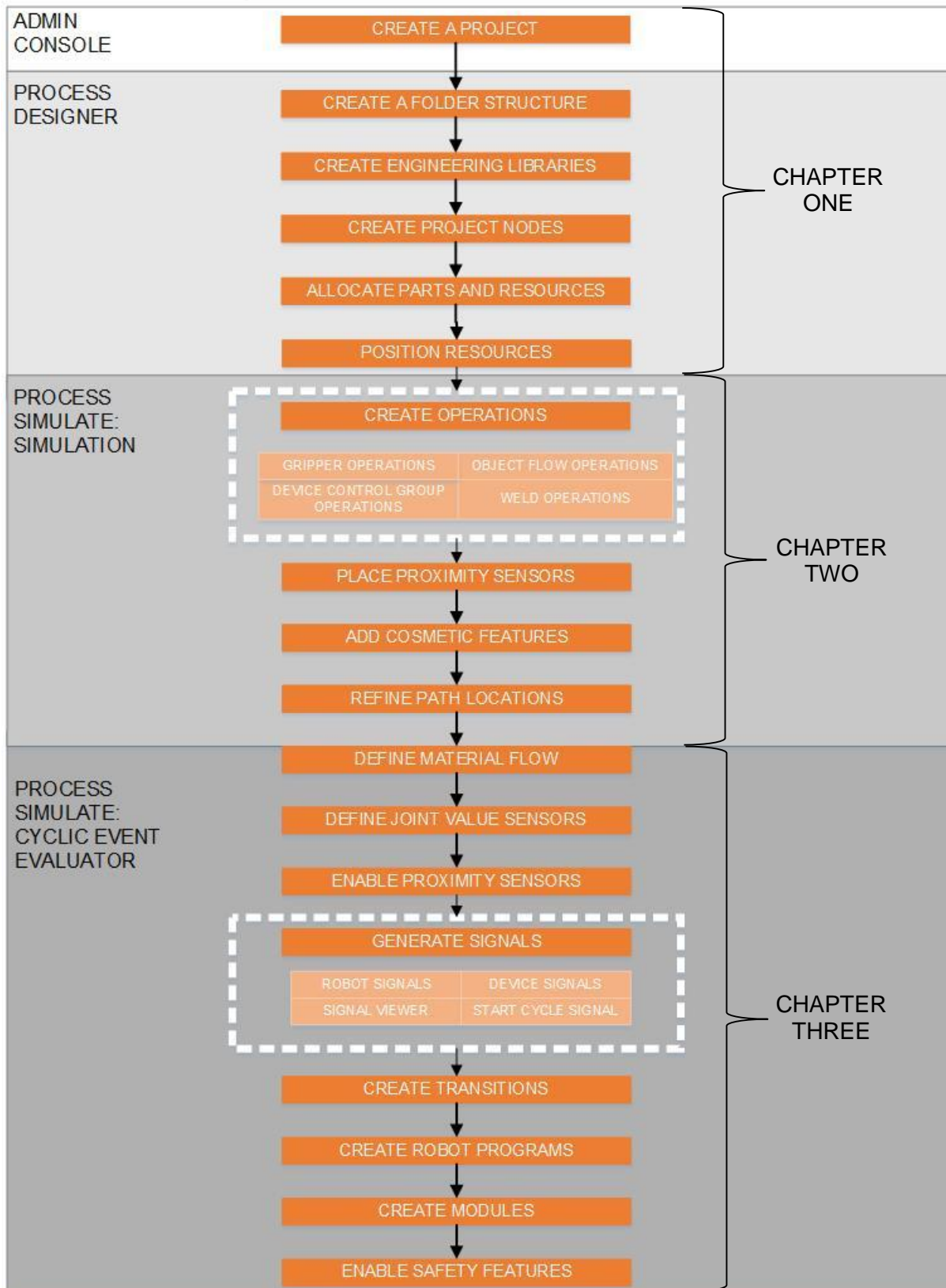
TUTORIAL SET-UP

This tutorial is designed to instruct a user in creating the virtual factory, shown in the diagram below. All steps necessary will be described in detail and it is hoped that through completing this tutorial, a user will be supplied with an introductory knowledge into the application of Virtual Commissioning using the Process Designer/Process Simulate software packages.



The user will be instructed in the creation of robotic and device operations as well as gaining a deep understanding of the extent to which Process Simulate can be used to test PLC logic. This function known as the Cyclic Event Evaluator in Process Simulate is a valuable tool used within industry during the Virtual Commissioning process. Due to the large number of stages involved in creating the virtual factory, this tutorial has been split into three chapters, each focusing on different aspects of the Virtual Commissioning procedure. Chapter one introduces a basic foundation using both Process Designer and Process Simulate, where Resources will be placed. Chapter two will then introduce the use of Operations within Process Simulate while Chapter three will conclude with the functionality offered by the Cyclic Event Evaluator. The flow chart displayed on the page opposite provides an overview of the steps within this tutorial and can be applied to the development of most virtual factories.

VIRTUAL FACTORY CREATION





CHAPTER ONE - INTRODUCTION

1.0 SOFTWARE OVERVIEW

The Tecnomatix Suite is comprised of two software packages used within this tutorial. Firstly is the Process Designer package where the folder structure for the project is created and Resources are imported. Secondly is the Process Simulate package where positioning of Resources takes place followed by, the addition of Operations and subsequently the implementation of logic. A more detailed description is provided below.

1.1 PROCESS DESIGNER

Within Process Designer, the libraries of CAD Parts and internal 'nodes' are defined and created. This part of the software suite allows the user to establish the library structure used for the remainder of the project and within industrial projects should follow the working method supplied by the organisation. In addition to this, Process Designer is referred to at various stages of the project and is used, for example, to define the associations between Operations, Resources and Parts. In essence, Process Designer is used to provide the backbone to the process before entering Process Simulate where the further functions are used.

1.2 PROCESS SIMULATE

Process Simulate is the primary software utilised in the development of an event-based simulation and is used to define the Operations and locations of each Resource and Part. In addition 'via points' and further sequences are defined in this software. When creating the event-based simulation, the Cyclic Event Evaluator (CEE) functions are then utilised to apply logic to the simulation in such a way that it can run in a cyclic manner based on user defined logical commands.

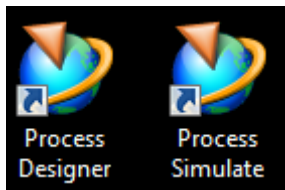
1.3 THE TECNOMATIX SETUP

In using both Process Simulate and Process Designer, a link between the two software packages is established through the means of an 'eMServer'. This server is often a standalone machine which contains the large amounts of data (e.g. CAD files) required for any given simulation. Communication between a machine and the 'eMServer' allows for multiple machines to operate within the server at the same time, securing projects through the use of a 'Check-Out/ Check-In' system.

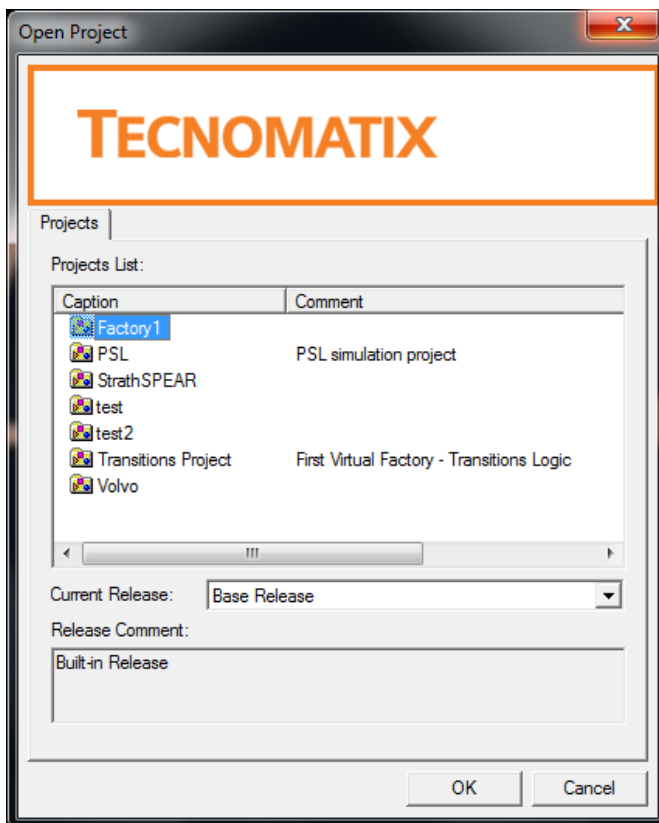
2.0 BASIC DEFINITIONS

2.1 OPENING THE SOFTWARE

Process Designer and Process Simulate are opened by selecting the desktop icons or start-menu icons shown below.



Upon opening, the user will be prompted to supply appropriate login details and select a project from the window displayed below.



The selected software package will then open with the chosen project already loaded within.

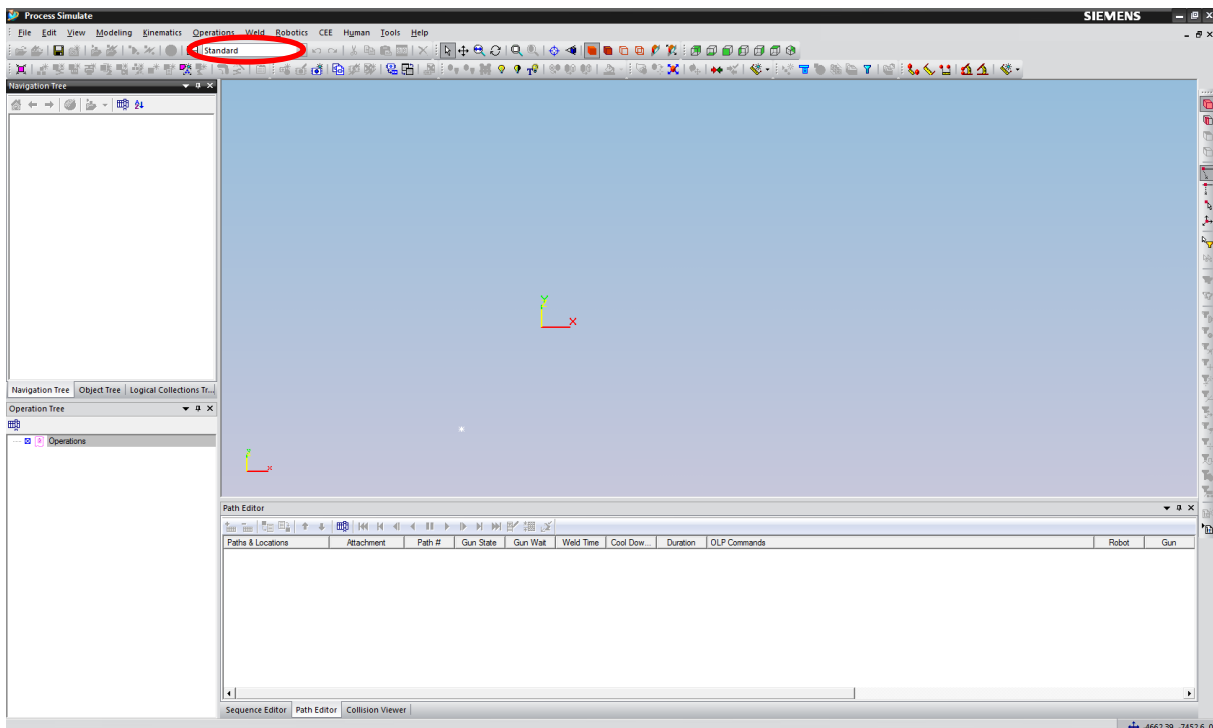
Once the software is open it is possible to open and close projects from within the software.

To close: File >> Close Project

To open: File >> Open Project – this will portray the list of projects available for opening.

2.2 VIEWERS

In navigating the software, 'viewers' are used and can be customised based on the users' needs by following View >> Viewers and selecting the desired viewer. Each viewer serves an individual purpose in the generation and evaluation of simulations. It is possible to select a default setup for the viewers based on the stage of the simulation creation. This will open the same default viewers in the same location every time. For the purpose of this tutorial, the 'Standard' setup will be sufficient for a large portion of the factory development and so this should be selected as shown below. The 'Standard' setup includes essential viewers such as the 'Navigation Tree', 'Operation Tree' and 'Graphics Window'. At a later stage in the factory development, including work within 'Line Simulation' mode, the 'Advanced Simulation' setup will be more appropriate.



2.3 NODES

Process Designer and Process Simulate work on the basis of 'nodes' within a folder tree. Each node can represent a Resource, Operation, Part or Manufacturing Feature. A colour key exists within the software and is as follows:

- Resources (**Blue**); Robots, Guns, Grippers, Fixtures etc.
- Operations (**Pink**); Robot Operations, Weld Operations etc.
- Parts (**Orange**); Car Parts
- Manufacturing Features (**Green**); Weld Points, Glue paths etc.

Nodes are created in the 'Navigation Tree' by right-clicking on the parent folder/node and selecting New >> Select the desired node.

2.4 USING THE MOUSE


Mouse functions are used extensively in Process Simulate. Within the 'Graphics Viewer' of both Process Simulate and Process Designer:

- Holding down the left mouse button will initiate a box which can be dragged over objects for selection.
- Using the middle button on the mouse facilitates zooming in and out of the graphics window.
- Holding down the middle mouse button and moving right zooms in, with zooming out being enabled by moving left.
- Using the middle and right mouse button together allow the 3D navigation of the models loaded within the 'Graphics Viewer'.

2.5 SYSTEM ROOT FOLDER

The 'System Root' is the central location that Process Simulate and Process Designer use for storing data and is where all files used/created within a simulation are stored. It is important not to delete any files connected to an existing/current simulation, else the simulation may become corrupt. The folder is typically located at 'C:\Tecnomatix\SystemRoot' however this may vary depending on your installation. Time should be taken now to locate your System Root folder.

2.6 SAVING YOUR SIMULATION

As you progress with any simulation project, you will need to frequently save your work to avoid the loss of any data. When working with an 'eMServer' this is done by selecting the 'Selectively Update' 'eMServer' button.  This saves any changes made to the simulation to the 'eMServer' and allows you to successfully store your data. It will be necessary to perform this step in both Process Designer and Process Simulate. In process Designer however, the save feature is labelled as 'Save Scenario' and will save the changes made to any nodes.

2.7 IMPORT/EXPORT OF SIMULATIONS

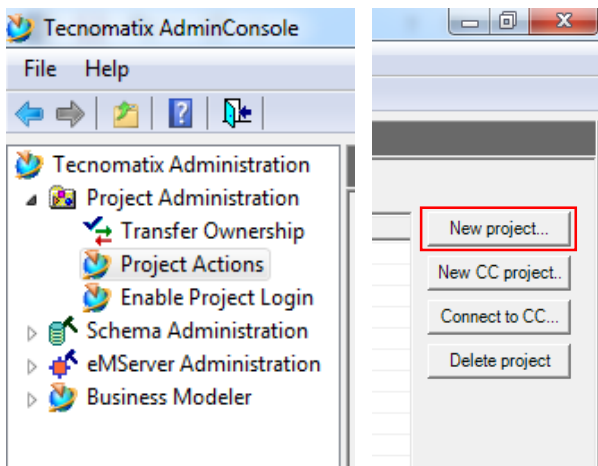
Alternatively, projects may be imported through the 'Pack and Go Import' function. At any stage in a project it is possible to save and export all work done as a 'Pack and Go export'. This will take all data within a study and saves it for import at a later date or on a different machine. Following File >> Project Management >> Pack and Go Import, the data is able to be re loaded within Process Simulate.

3.0 PRE-OPENING TASKS

1. All CAD files required for the simulation (Parts, Resources, Robots, Fixtures etc.) should be located in their .cojt format. For this tutorial, the folder is named '*Factory 2 CAD Parts*'.
2. Copy and paste the '*Factory 2 CAD Parts*' folder containing all of the files into the System Root folder.

4.0 CREATING A PROJECT

1. Open the Tecnomatix 'AdminConsole' via the Windows Start menu. Click 'Project Actions' then 'New Project'. Name the project and set comments as you wish. Close the 'AdminConsole'.

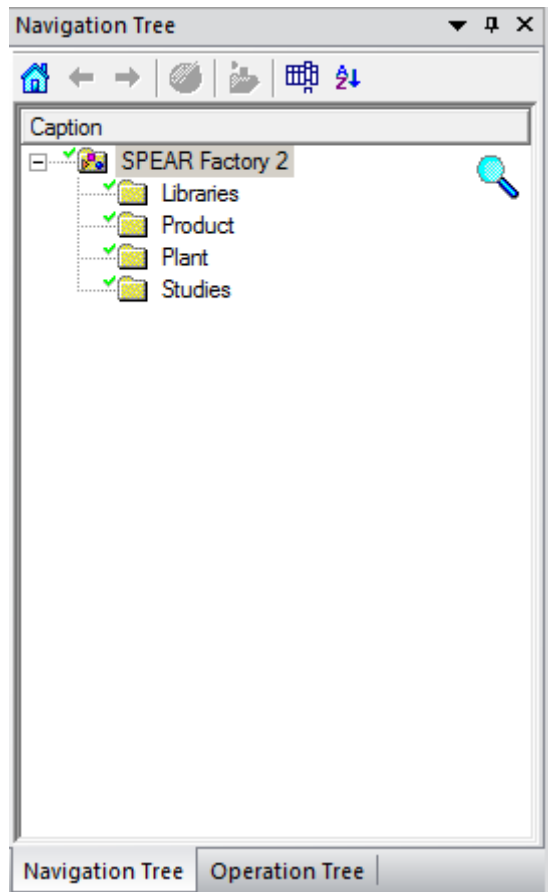


2. Open 'Process Designer' and select the factory created in step 1.
3. Check out the project. To do so, right-click the project and select 'Check Out'. Tick the 'Check out with hierarchy option'.

Note: The Check-in/out process is a method used within larger organisations to ensure that data is maintained securely. Within large projects it may be desirable to have more than one operator working on the project and so the check-in/out process is used to ensure that only one operator may edit particular parts of the project at a time. Once a node or folder has been checked-out it is only usable by that operator.

5.0 CREATING THE FOLDER STRUCTURE

1. Right-click on your factory node (named 'SPEAR Factory 2' in this tutorial) and click 'New'. Select 'Collection' and set the amount to '4' to create four new folders. Once created, expand the menu and click a folder. Press F2 to rename and repeat for each folder to match the structure shown below.



Note: These folders have the following functions:

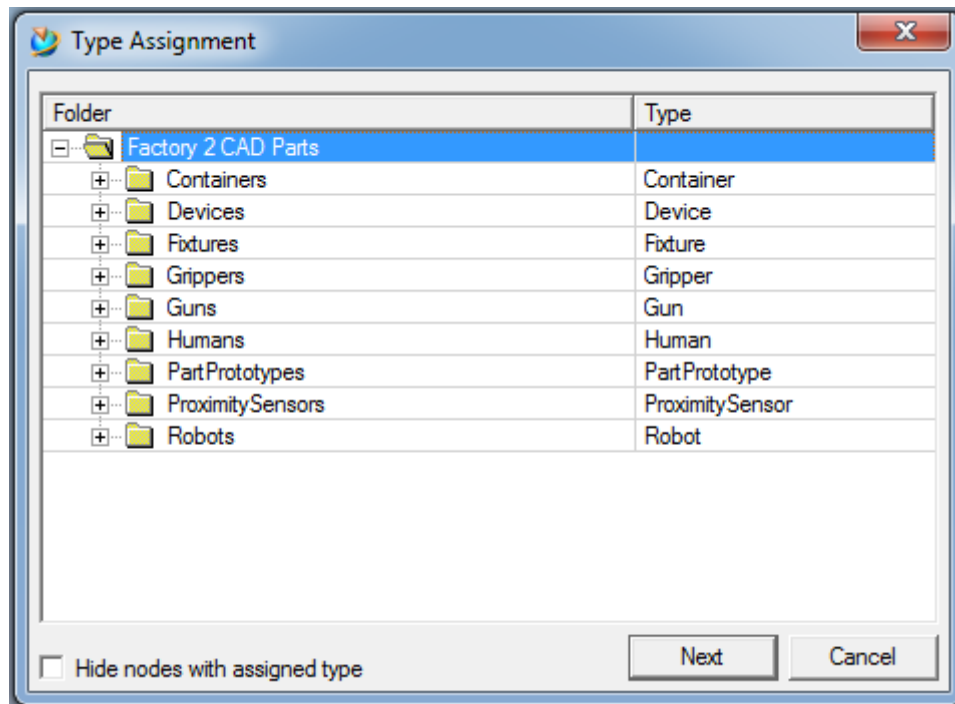
- **Libraries;** will hold all the 'EngineeringLibraries' that will be imported and effectively stores all the CAD data for the project.
- **Product;** contains all components associated with the parts being assembled in the factory
- **Plant;** contains location and resource allocation for Operations and Stations
- **Studies;** contains the 'Study' information such as 'RobCad' Studies

While these folders include the required information for this project, more may be needed for other projects.

6.0 CREATING ENGINEERING LIBRARIES

1. Follow the path Tools >> Administrative Tools >> Create Engineering Libraries. When prompted, select the folder previously created containing all CAD Parts located in the System Root (see Section 2.5).
2. Once selected, each CAD Part / Folder should show up individually. Each part now needs assigned a category based on its use within the simulation. For simplicity, the CAD Parts

in this tutorial have been grouped as per their intended use however time should be taken to explore the different categories available. Select the 'Type' to match as shown:



3. Click 'Next' to create the library. Drag and drop the new nodes named 'EngineeringResourceLibrary' and 'EngineeringPartsLibrary' into the 'Libraries' folder.

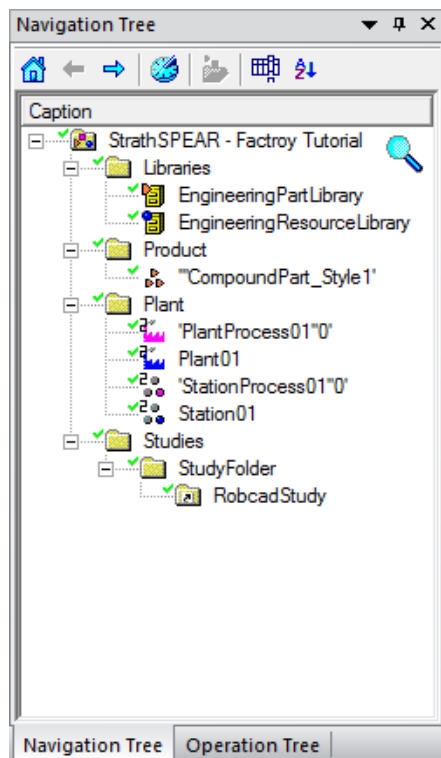
7.0 CREATING THE PROJECT NODES

1. Create a 'Compound Part' within the 'Product' folder renaming this node as 'CompoundPart_Style1'. To do so, right-click the folder, click 'New' and select 'Compound Part' from the available node types.
2. Create a 'PrPlant' and 'PrStation' within the 'Plant' folder following the same procedure as above.

Note: Once created, two additional nodes will be generated as 'PrPlant' and 'PrStation' nodes come in pairs. For every Resource 'PrPlant' (blue) node, there is an associated Operations 'PrPlant' (Pink) node and the same is true of 'Stations'.

3. Create a 'StudyFolder' within the 'Studies' folder. This allows the user to create a variety of the different studies available within the software. For the purpose of this tutorial, create a 'Robcad Study' within the 'StudyFolder'.

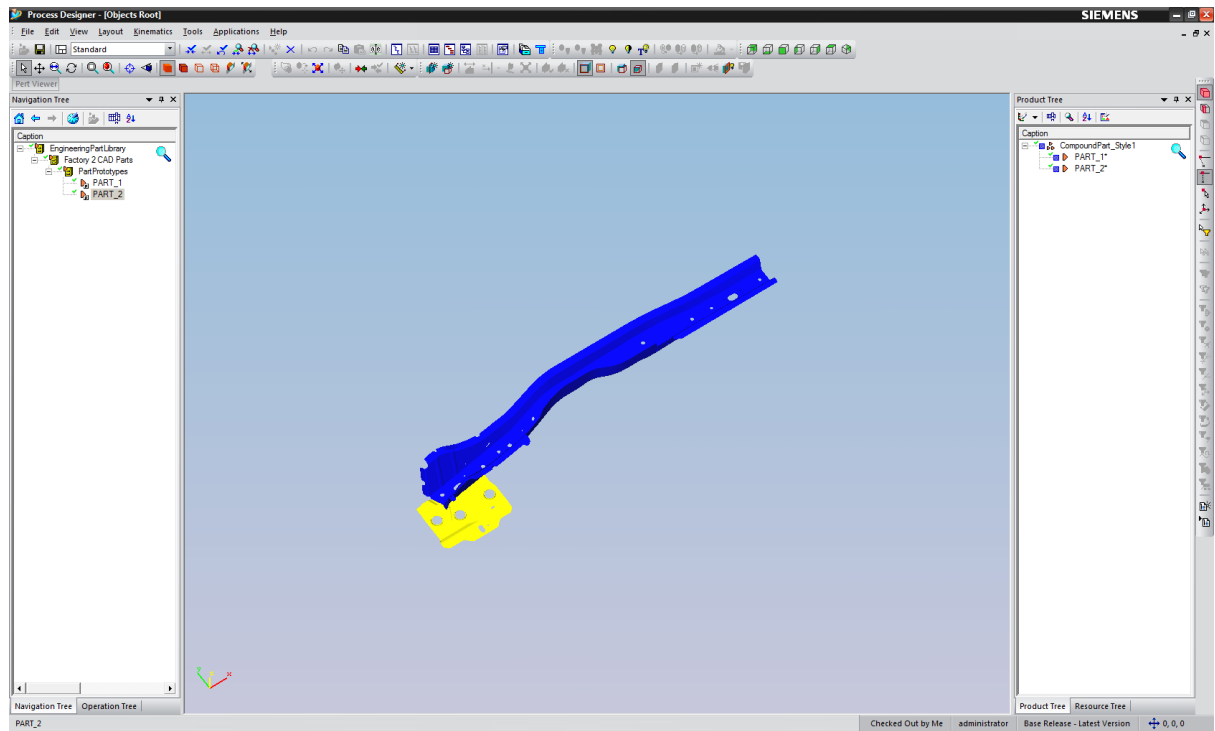
- Your final folder and node structure should look like the diagram shown below. Rename any nodes as necessary by clicking the node once and pressing F2.



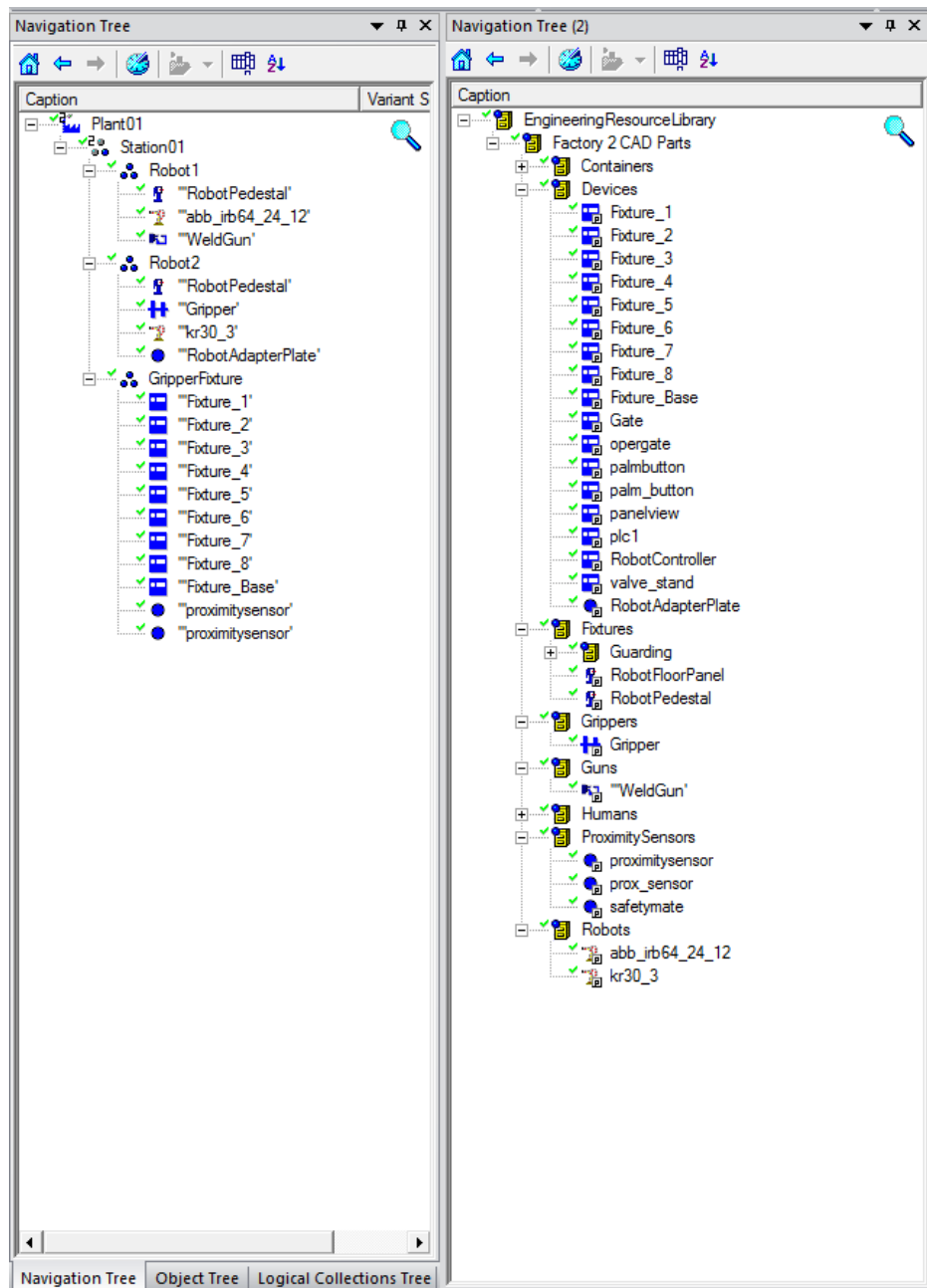
- Click and drag the 'StationProcess01'0' node into the 'PlantProcess01'0' node (the corresponding 'Station01' will move automatically due to the reasoning explained above). This will generate the standard hierarchy where Plant is the top level and Station is a sub-node of the Plant.
- Prior to the allocation of Parts and Resources, it is good practice to group Resources within a compound resource node. As such, for this tutorial, create three 'CompoundResource' nodes within 'Station01'. Rename as 'Robot1', 'Robot2' and 'Fixture'.

8.0 ALLOCATING PARTS AND RESOURCES

- To allocate the parts that will be used in the tutorial, right-click the 'CompoundPart_Style1' node and select 'Load' (ensuring the 'Product Tree' viewer is open prior to doing so; View >> Viewers >> Product Tree). Drag Parts 1 and 2 into the 'CompoundPart_Style1' node shown in the 'Product Tree'. Your setup should look like that shown below.



2. To allocate Resources, it is first necessary to open two 'Navigation Trees' side-by-side as shown below; this will assist in dragging and dropping Resources into the correct nodes. Right-click on 'EngineeringResourceLibrary' and select 'Navigation Tree' to create the layout as shown.

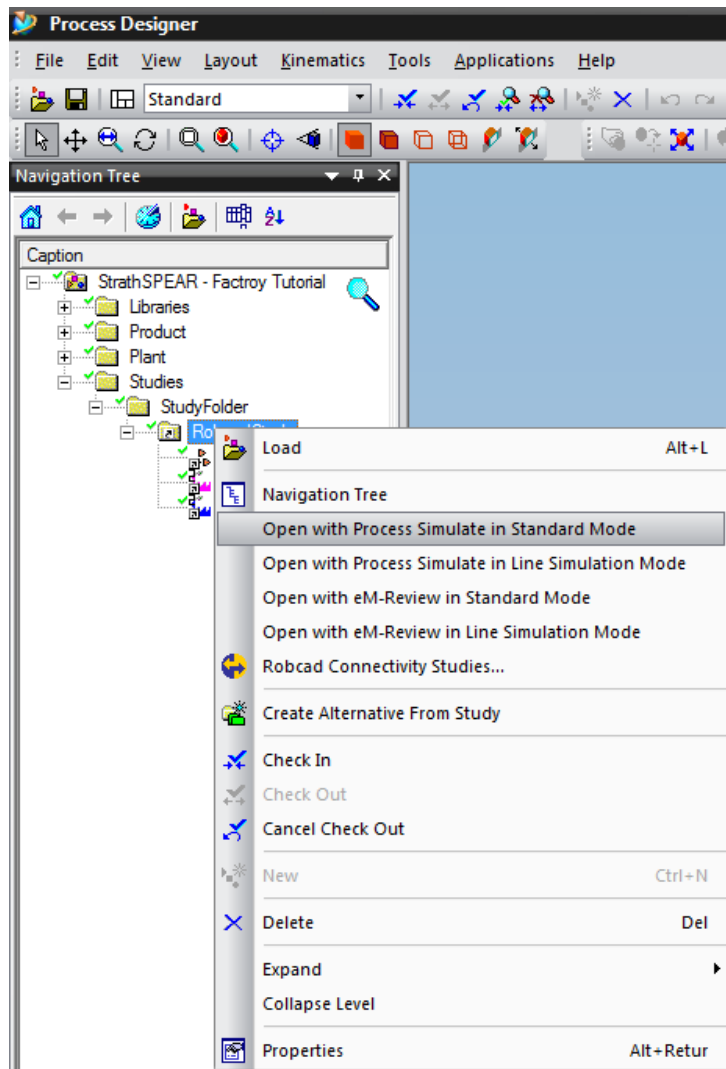


3. Click and drag Resources from the 'Navigation Tree' on the right to the correct nodes in the 'Navigation Tree' on the left. Your final setup should resemble that as displayed.
4. Click and drag the three nodes ('CompoundPart_Style1', 'PlantProcess01"0' and 'Plant01') into the previously created 'Robcad Study'. This creates a shortcut for each node within the 'Robcad Study' and allows the study to reference these parts.
5. Save the scenario (File >> Save Scenario). This completes the setup of the project within Process Designer and the following actions will be carried out in Process Simulate.

Note: Cosmetic and safety features will be added at a later stage in this tutorial as these can be created directly within Process Simulate. See Section 13.

9.0 OPENING IN PROCESS SIMULATE

1. Right-click the 'Robcad Study' node and select 'Open with Process Simulate in Standard Mode' as shown.

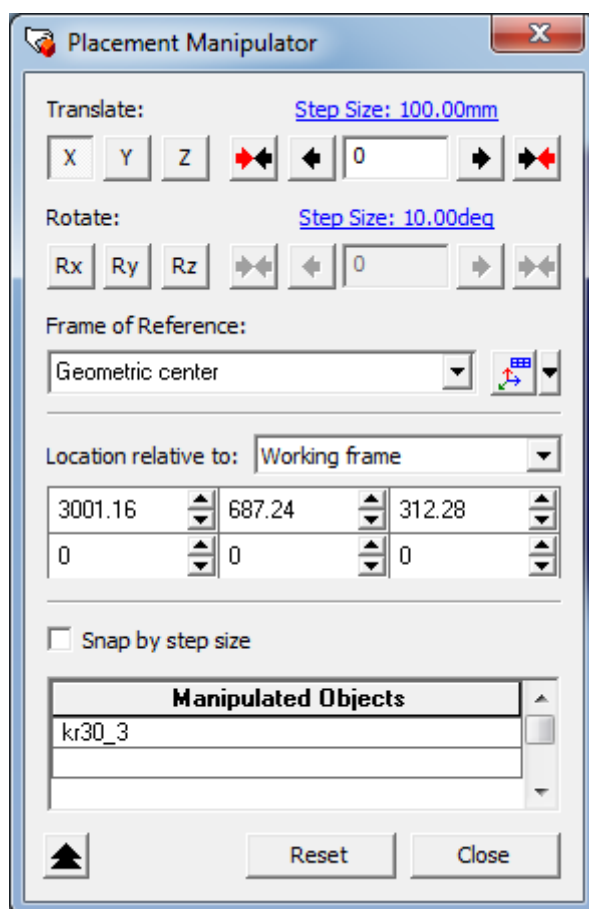


2. It is useful to display the floor of the factory to give a clearer visual representation. Click View >> Display Floor.
3. In manipulating objects, it is beneficial to alter their position relative to the 'Working Frame'. This should be set by clicking Modeling >> Set Working Frame and selecting 'Reset to Origin'.

10.0 POSITIONING OF RESOURCES

In Process Simulate, all Resources will be displayed in the 'Graphics Viewer' but it is also necessary to specify positions for each resource individually, representative of the desired factory layout. One method is to right-click the resource and select 'Placement Manipulator'. This opens a frame by which the resource can be dragged and rotated to the correct position. Alternatively, and for increased accuracy, co-ordinate points can be input by the user.

1. Firstly, select the 'kr30_3' Robot and open up the 'Placement Manipulator'. Enter co-ordinate points as follows:

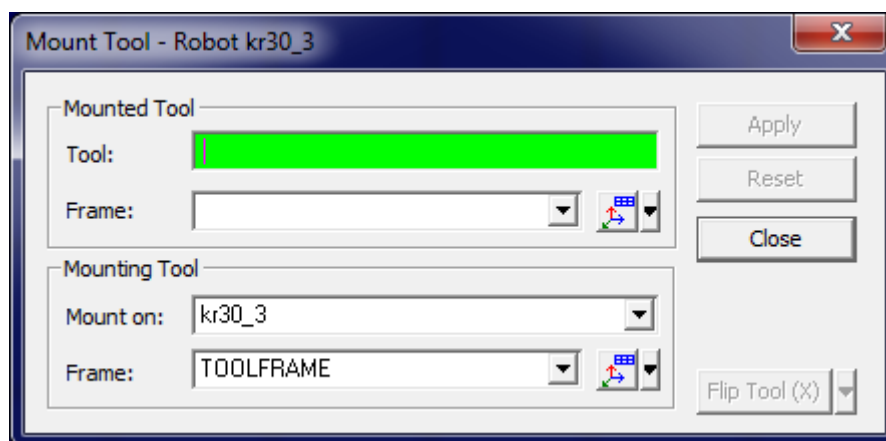


- Repeat this process for all Parts as shown in the table below.

	X	Y	Z	Rx	Ry	Rz
RobotPedestal	2982.76	721.84	0	0	0	0
RobotPedestal	7835.54	260.52	0	0	0	0
abb_irb_64	7795.06	267.02	295.13	0	0	180

Note: Please enter the Rz co-ordinate first for the 'abb_irb_64' Robot—this will ensure that the absolute position is correct.

- With Resources in place, it is necessary to mount tools onto their associated Robots. Do so by right-clicking the Robot and selecting 'Mount Tool'. This will open the following dialogue window:



Firstly, complete this step for the 'kr30_3' Robot selecting 'RobotAdapterPlate' as Tool and 'Self' as the mounting frame to complete the initial mounting procedure.

- Again, selecting the 'kr30_3' Robot, open the 'Mount Tool' command box and this time select the 'Gripper' as the mounted tool. In this instance, the mounting frame should be set as 'FR1' and the orientation should be adjusted with a 90° shift in the Z-axis using the 'Flip Tool'.
- Following this, select the 'abb_irb64' Robot and open the 'Mount Tool' command box. Mount the 'WeldGun' using the 'mtg' mounting frame, again with a shift of 90° in the Z-axis.

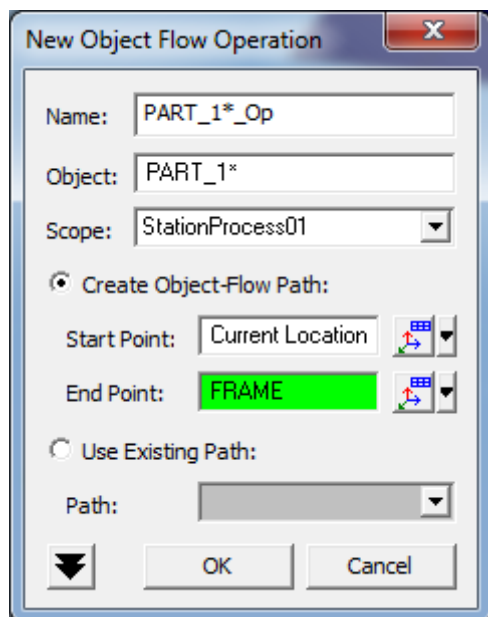
Note: Once the tools are mounted it is possible to test that they function as desired through the 'RobotJog' function. Right-click the desired Robot and select 'RobotJog'. By manipulating the position of the Robot you will be able to see if the tool moves alongside the Robot, confirming the functionality of the Robot.

CHAPTER 2 – CREATING OPERATIONS

11.0 CREATING OPERATIONS

11.1 OBJECT FLOW OPERATIONS

1. Double-click 'CompoundPart_Style1' in the 'Navigation Tree' to display the individual parts. Selecting 'PART_1', click Operations >> New Operation >> New Object Flow Operation. This Operation is used to simulate a part flowing along a pre-defined path and in the case of this tutorial, it will emulate the action of a human placing both parts into the Clamp Fixture. The following dialogue box will appear:



Set the 'Scope' as shown by selecting the node in the 'Operation Tree'. Set the 'End Point' by clicking in the entry-box to turn it green, then selecting an arbitrary point within the 'Graphic Viewer'. These points will be updated with precise co-ordinates at a later stage and so it is not necessary to define points at this stage.

2. Repeat the above procedure for 'PART_2' creating two 'Object Flow Operations' within the node 'StationProcess01'.
3. It is beneficial to rename all Operations as they are created, particularly for latter stages in the project when signals are defined. As such, by selecting the node in the 'Operation Tree' and using F2, rename as 'PART_1_to_Fixture' and 'PART_2_to_Fixture' accordingly.
4. Selecting 'StationProcess01' in the 'Operation Tree' will allow the Operations to be added to the "Path Editor", either via a drag and drop action or by selecting the 'Add Operations to Editor' icon highlighted in the top left of the "Path Editor" viewer. Now enter the co-ordinates as shown below:

Path Editor

Paths & Locations	Attachment	X	Y	Z	RX	RY	RZ
StationProcess01							
PART_1_to_Fixture							
loc		1660.45	-7194.93	952.60	0.00	0.00	180.00
loc1		4448.34	-620.41	952.60	0.00	0.00	180.00
PART_2_to_Fixture							
loc2		5119.00	-7522.44	815.59	0.00	0.00	180.00
loc3		5119.00	-501.46	815.59	0.00	0.00	180.00

Sequence Editor | Path Editor | Collision Viewer

- In reality, additional path locations are required to define the actual motion of the part. As such, further locations should be added to these 'Object Flow Operations'. Right-click 'loc1' within the "Path Editor" as shown above and select 'Add Location Before'. This will add an intermediate location to simulate the lowering of the part into the clamp. Complete this step for both 'Object Flow Operations'. Set the values as shown below in either the 'Placement Manipulator' or the "Path Editor" window. In addition, rename the locations as displayed.

Path Editor

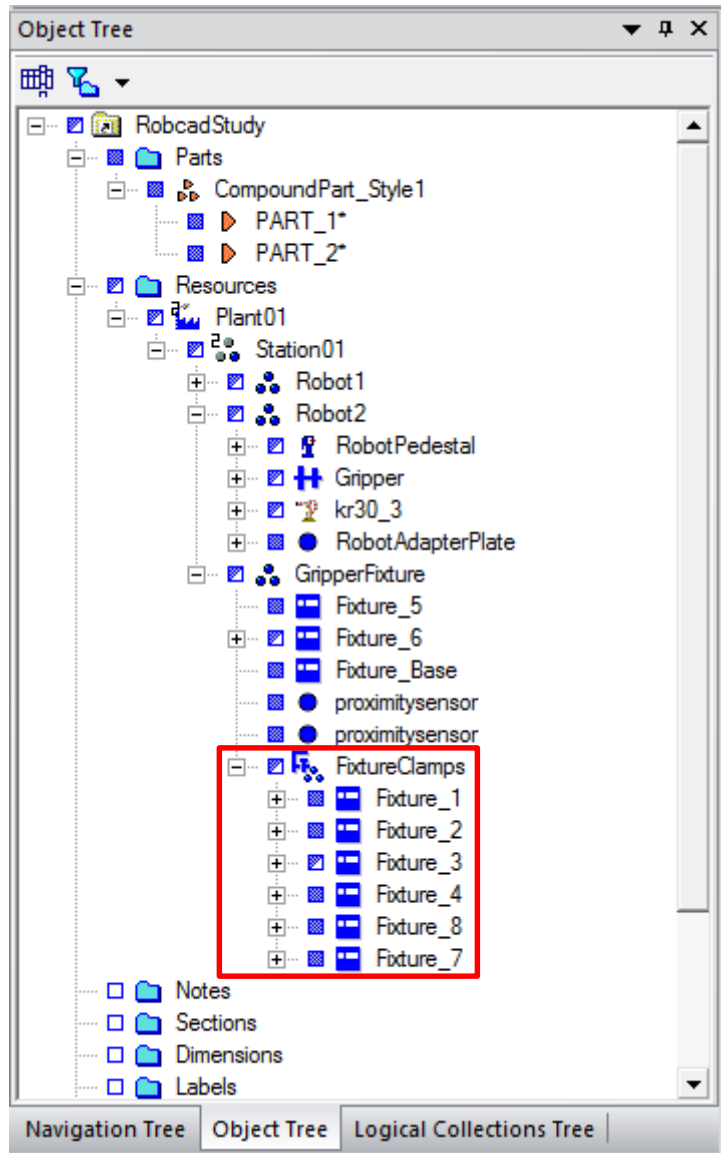
Paths & Locations	Attachment	X	Y	Z	RX	RY	RZ
StationProcess01							
PART_1_to_Fixture							
PART_1_loc_1		1660.45	-7194.93	952.60	0.00	0.00	180.00
PART_1_loc_2		4448.34	-620.41	1150.00	0.00	0.00	180.00
PART_1_loc_3		4448.34	-620.41	952.60	0.00	0.00	180.00
PART_2_to_Fixture							
PART_2_loc_1		5119.00	-7522.44	815.59	0.00	0.00	180.00
PART_2_loc_2		5119.00	-501.46	1015.00	0.00	0.00	180.00
PART_2_loc_3		5119.00	-501.46	815.59	0.00	0.00	180.00

Sequence Editor | Path Editor | Collision Viewer

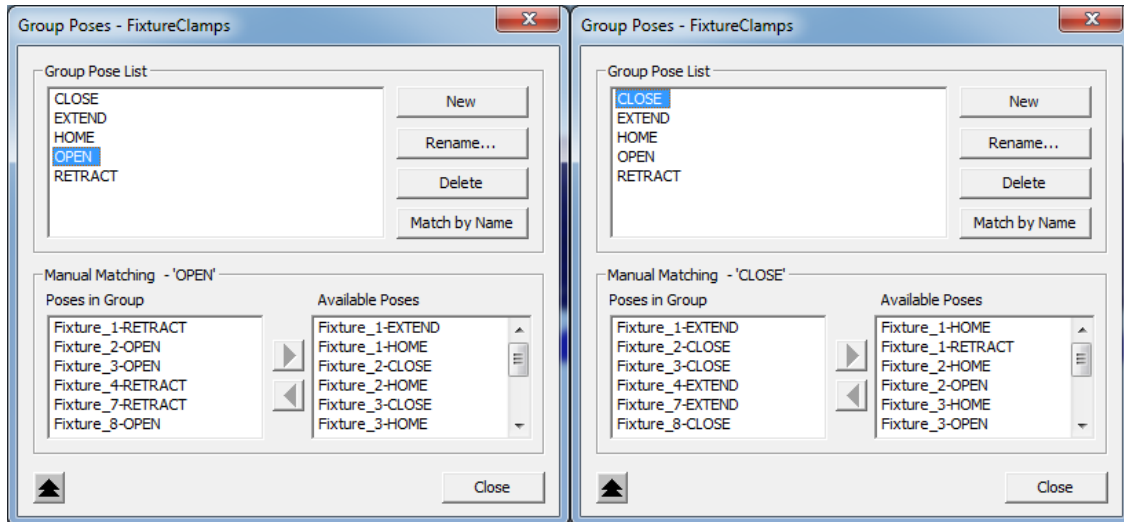
11.2 DEVICE CONTROL GROUP OPERATIONS

- At present, the above 'Object Flow Operation' is not accompanied by the necessary clamp Operations required to hold the parts in the Fixture. As such it is necessary to create a 'Device Control Group' which will allow all the clamps on the Fixture to be operated by one 'Device Control Group Operation'.
- Create a 'Device Control Group' by clicking CEE >> Device Control Group >> Create Device Control Group.

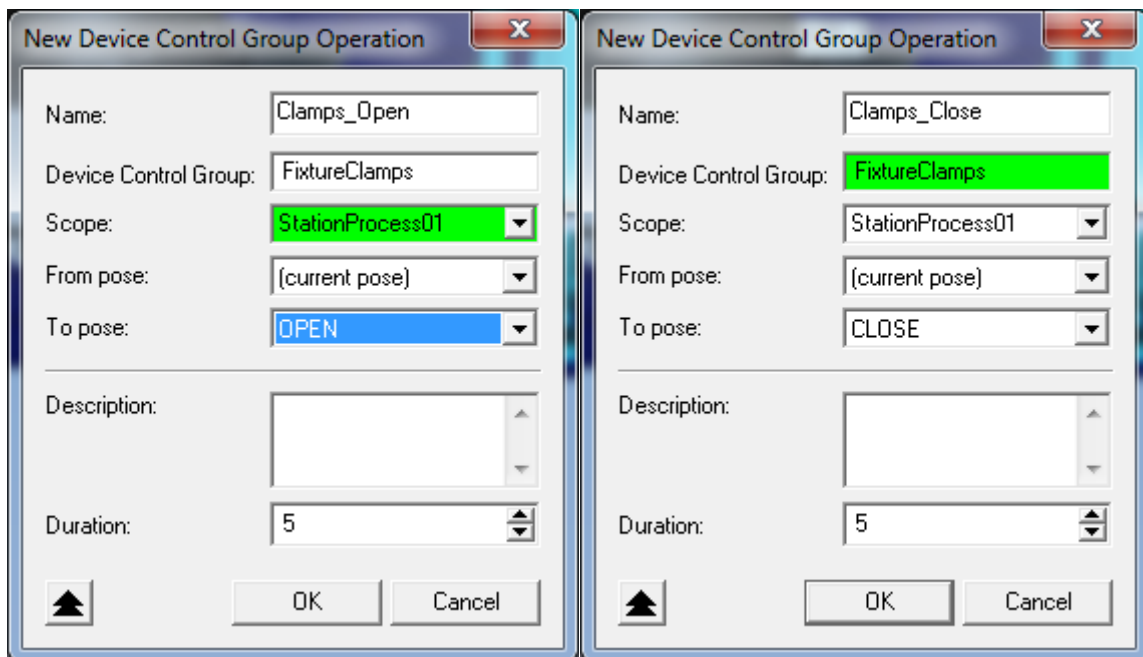
3. Navigating back to the 'Object Tree', under 'Resources' the newly created 'Device CG' can be found. Rename this node to 'FixtureClamps' and place it within the 'GripperFixture' node.
4. Using a drag and drop action, place the following clamps into the 'Device Control Group': Fixture 1, 2, 3, 4, 7, 8 as shown in the diagram below.



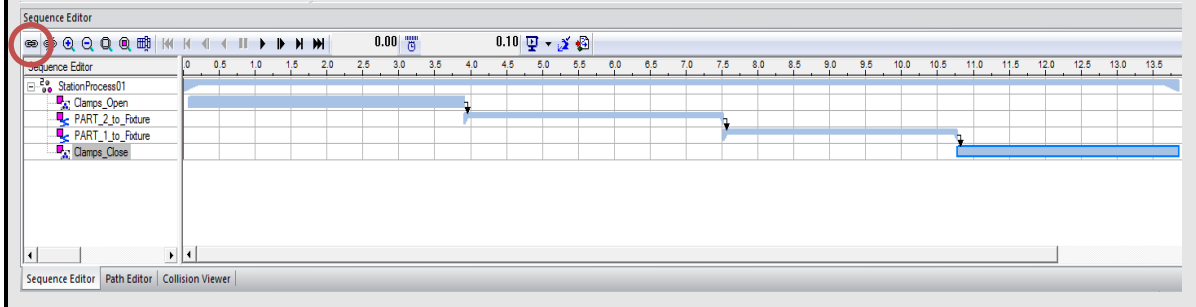
5. Selecting the 'FixtureClamps' control group, select CEE >> Device Control Group >> Edit Pose Groups. This function is used to associate the poses of the clamps with each other such that all Operations are performed in synchronisation. First, clicking 'Match by Name' then selecting 'CLOSE' from the 'Group Pose List', it is necessary to add 'EXTEND' for Fixtures 1, 4 and 7 to this group pose. Similarly for 'OPEN', Fixtures 1, 4 and 7 should have 'RETRACT' added to this group pose. See below.



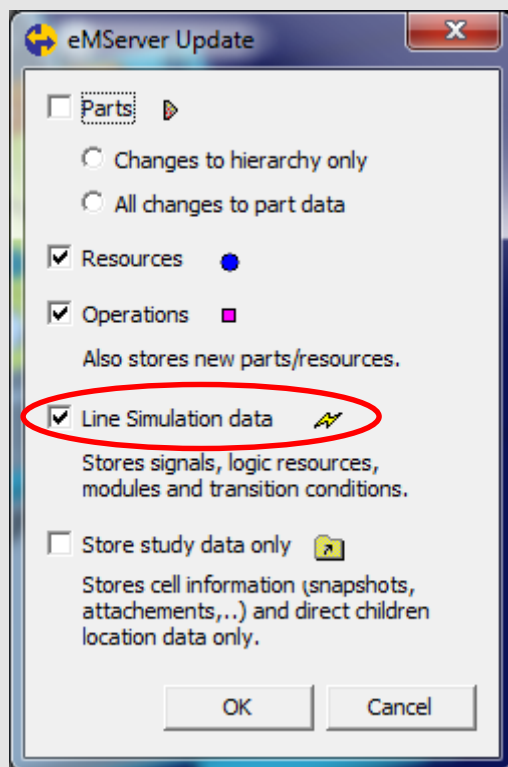
- The Device Control Group Operation can now be created. From the menu bar, click Operations >> New Operation >> New Device Control Group Operation. Rename this Operation to 'Clamps_Open' and select 'FixtureClamps' as the Device Control Group. As the Scope, select 'StationProcess01' and set the 'To pose' to 'OPEN'. Repeat this procedure for the 'CLOSE' Operation. See below.



Note: As you now have four Operations, it is necessary to adjust the order in which the Operations take place. Do so in the 'Sequence Editor' window by linking Operations such that the Gantt Chart correctly displays the order in which the Operations occur. Link Operations by highlighting the desired Operations (in the correct sequence) then select the 'Link' button circled in the top-left of the 'Sequence Editor' window. The correct sequence at this stage should appear as follows with black arrows indicating the links.



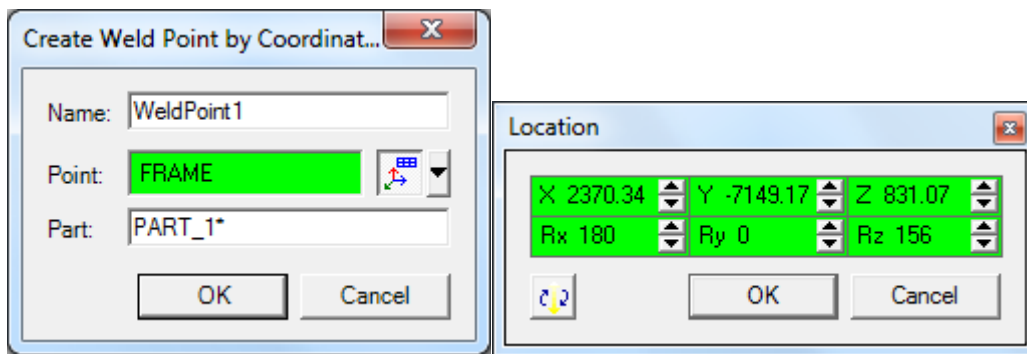
Note: When saving from now on, ensure that 'Line Simulation data' is ticked, or else the save will be incomplete!



11.3 WELD OPERATIONS

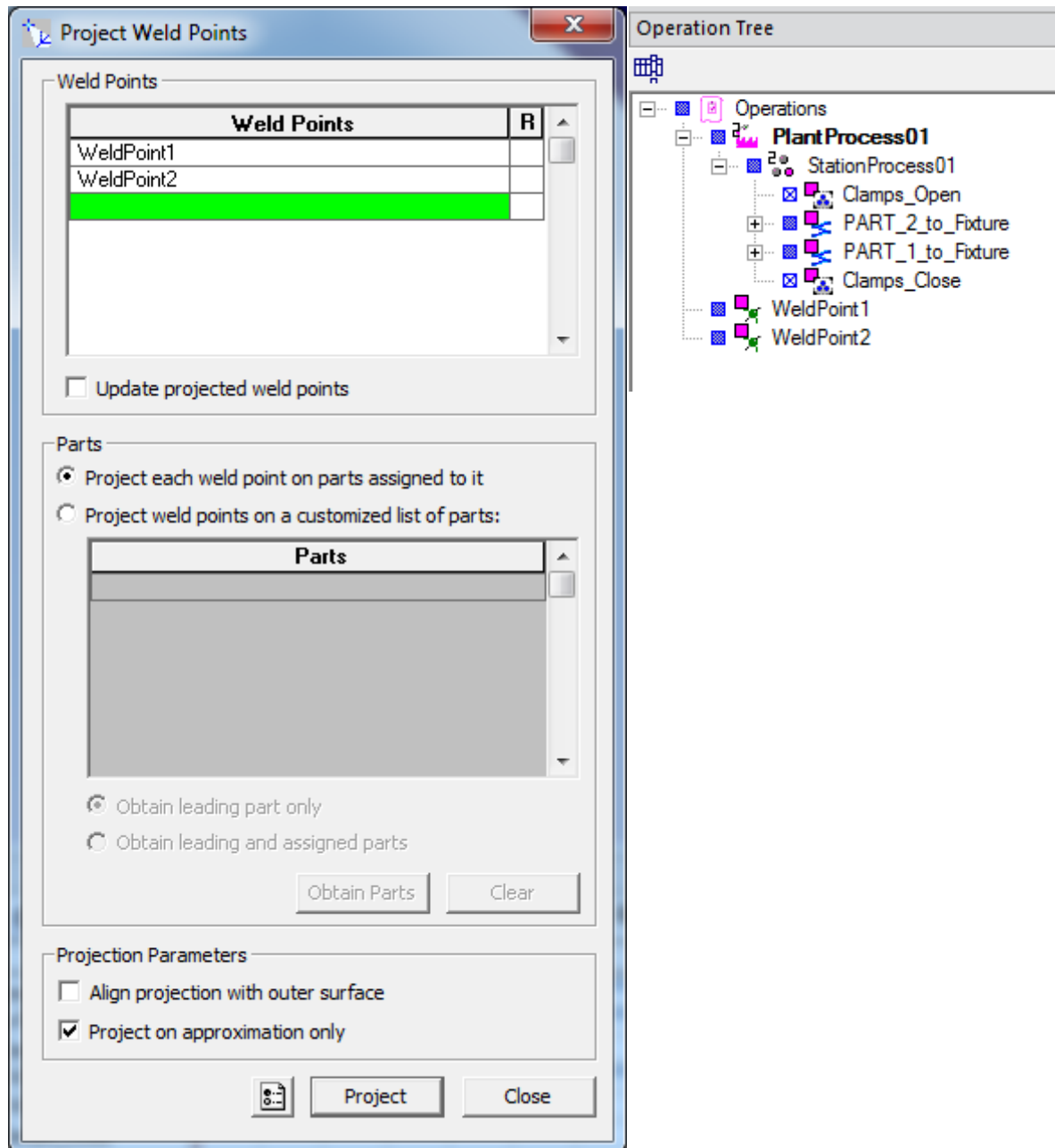
With the parts correctly secured in the Fixture it is now necessary to fix the parts together through a 'Spot Weld Operation'. Creation of a 'Spot Weld Operation' requires the definition of a Weld Point, which will become a 'Manufacturing Feature', and the subsequent projection of these points onto the parts. The following steps detail this process:

1. With the simulation reset, select: Weld >> Spot >> Create Weld Points By Coordinates. Rename this first Weld Point as 'WeldPoint1' and set a 'frame by 6 values' to the coordinates specified in the table below, selecting 'PART_1' as the part associated to this. The figure below shows an example of how to do this. Repeat this action for a second Weld Point again following the coordinates below.

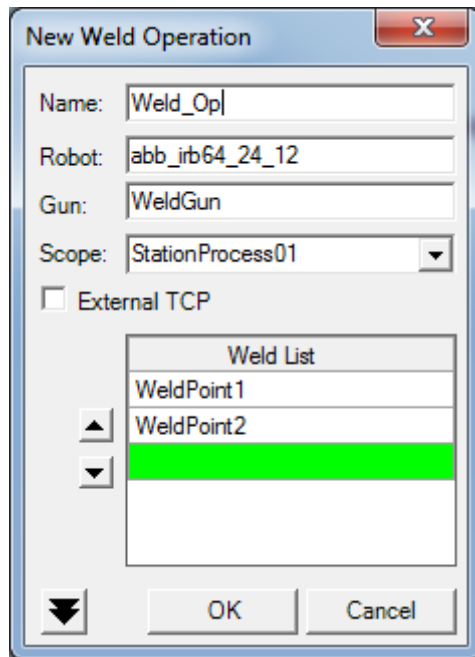


Weld Point	X	Y	X	Rx	Ry	Rz
WeldPoint1	2370.34	-7149.17	831.07	180	0	156
WeldPoint2	2323.65	-7153.21	831.07	0	0	160

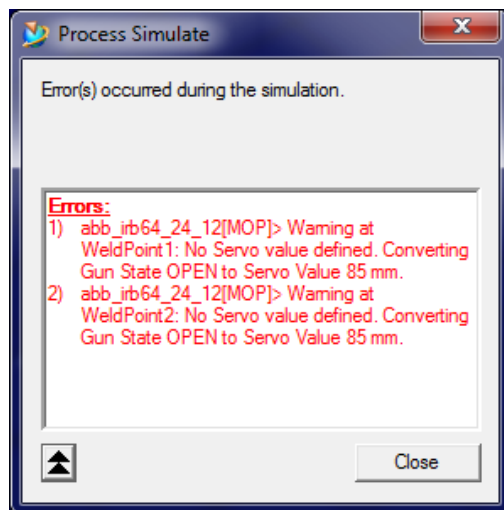
2. The next step is to project these Weld Points onto the defined part. Select Weld >> Spot >> Project Weld Points. In the newly opened dialogue box select 'WeldPoint1' and 'WeldPoint2' from the 'Operation Tree' and select 'Project' as shown in the diagram below. Following this the Weld Points will be projected and should be displayed as solid pink nodes in the 'Operation Tree'.



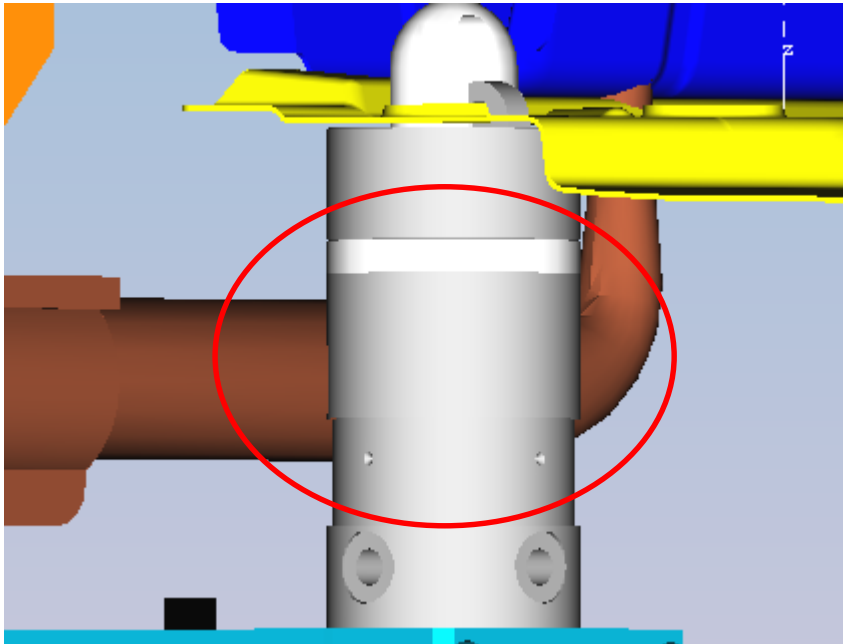
3. With the Weld Points created, the 'Weld Operation' will now be defined. Follow: Operations >> New Operation >> New Weld Operation to create this operation. A 'Weld Operation' dialogue box will now appear.
4. In the 'Robot' field, select the 'abb_irb64' Robot from the 'Graphics Viewer' and populate the 'Scope' field with 'StationProcess01'. In the 'Weld List' box, select 'WeldPoint1' and 'WeldPoint2' from the 'Operation Tree' as shown below.



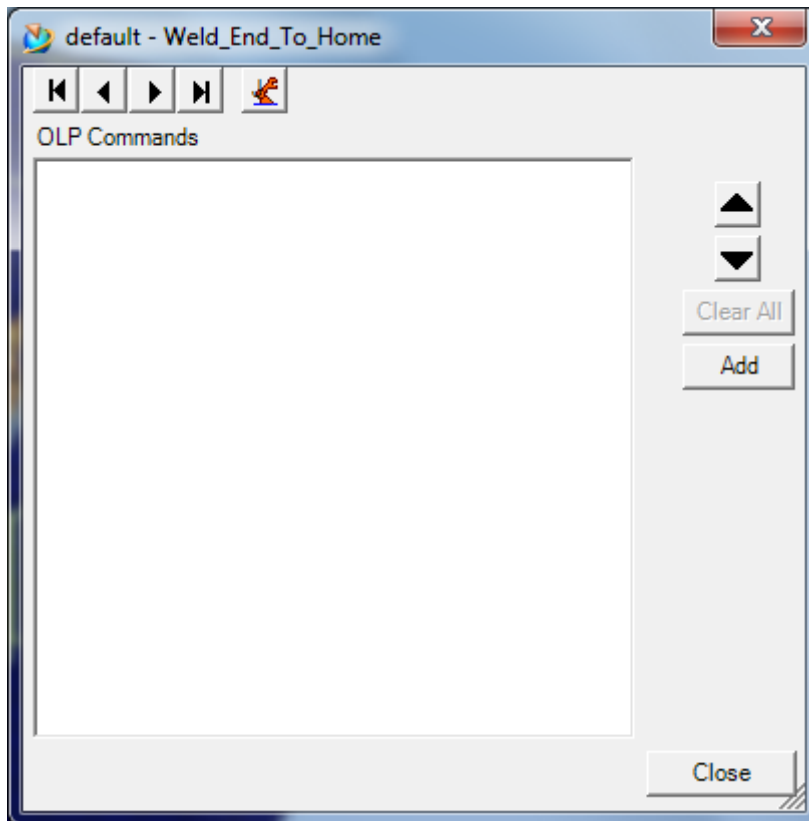
- As described previously, the sequence editor is used to define the order the events occur in. Link the 'Clamps_Close' Operation to the newly created 'Weld_Op' now and click play to view the simulation so far. Ignore the warning message that appears here, shown below:



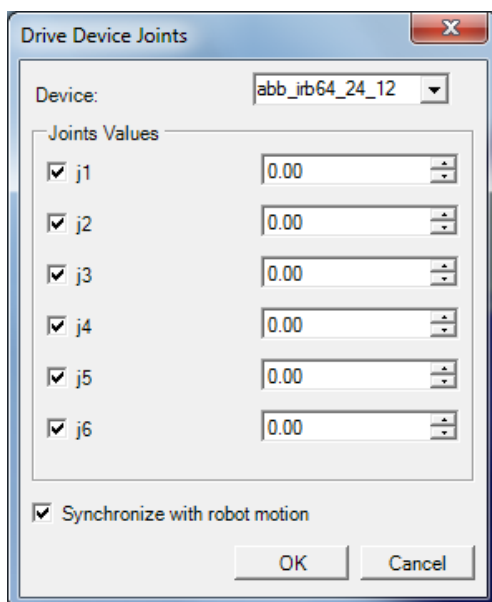
- Upon playing the simulation there is a clear collision between the Weld Robot and the Fixture as shown below. As such it is necessary to rotate the angle at which the Weld Robot will perform the weld.



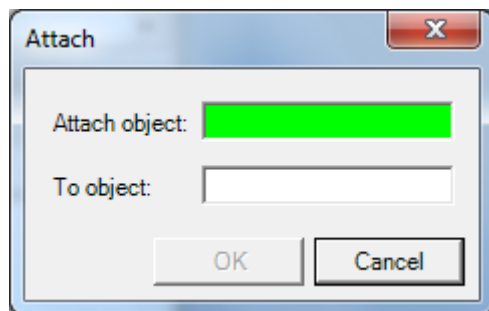
7. After letting the simulation run to completion, select each Weld Point in turn in the 'Operation Tree' and open: Weld >> Pie Chart to adjust the weld angle. Set the value of the 'Rotate Location Around Perpendicular' box to -20° . This angle will remove any collisions of the weld gun.
8. It can be noted that following the Operation, the Weld Robot does not return home. Adding in a 'via location' after the 'Weld Operation' will allow the simulation to run as required. Right click on the 'abb_irb64' Robot and select 'Home'. Following this, open the "Path Editor" and within the 'Weld_Op' right click 'WeldPoint2' and select 'Add Current Location'. This will add the current Robot location (Home) to the list of points within the 'Weld Operation'. Ensuring this point is the last in the list, and renaming the point to 'Weld_End_To_Home', will send the Robot to Home after the 'Weld Operation' is complete.
9. As an additional safety measure to ensure the Robot definitely returns to both the home position and configuration, an 'OLP command' can be used. Loading 'Weld_Op' in the "Path Editor" select the box named 'OLP Commands' for the 'Weld_End_To_Home'. The window shown below will then be displayed.



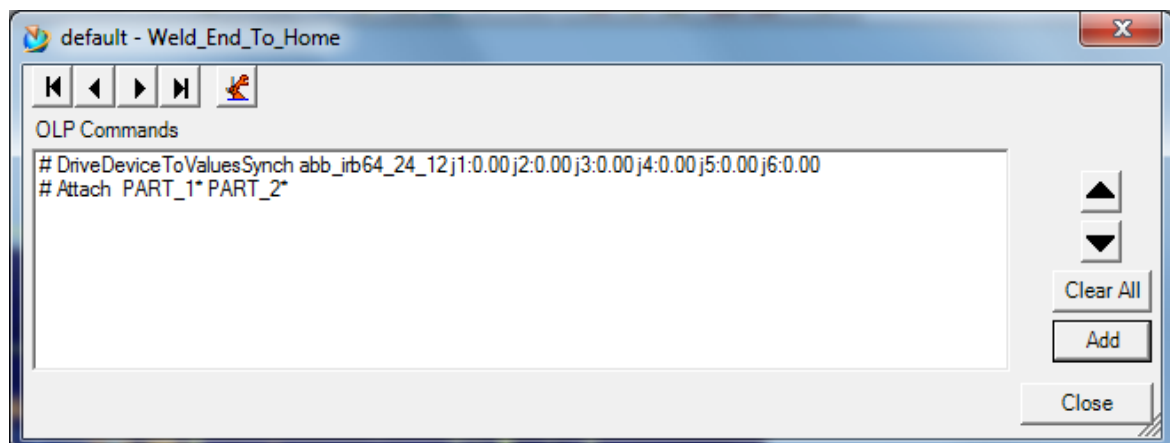
10. Selecting the 'Add' button, follow 'Standard Commands >> Tool Handling >> Drive Device Joints'. This will then open a second window shown below with all data filled in. The window which opens will give the option to select a 'Device' select the 'abb_irb64' Robot now. This feature will drive the Robot to the joint configuration selected. As the original 'HOME' pose has a joint configuration of '0' for each joint, select each joint in turn and leave the value as '0'. In addition select the box 'Synchronize With Robot Motion'. Then select 'OK'.



11. In addition to supplying the above 'OLP command' to ensure the Robot returns home, it is necessary to input a command which specifies to the software the two parts being welded should now act as one part. This is again done in the 'OLP Command' window. As before, select the OLP Command window on the 'Weld_Op_End_To_Home'. Now select 'Add >> Standard Commands >> Part Handling >> Attach' to display a new window as shown below.



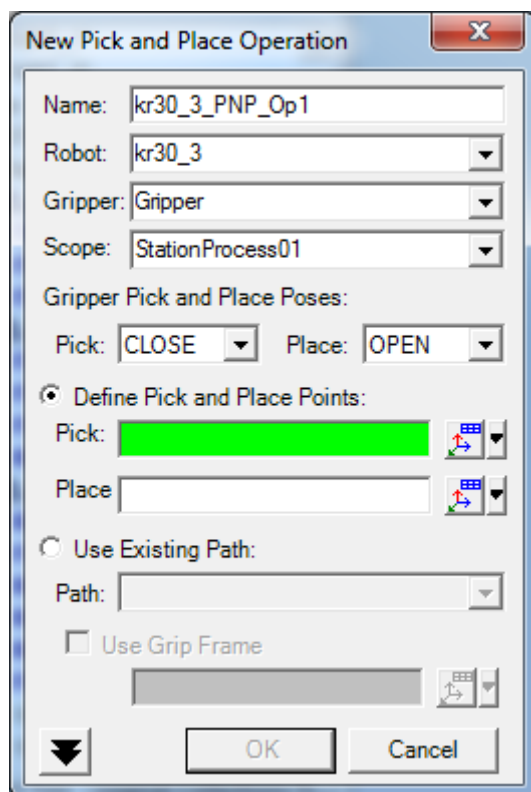
12. In the 'Attach Object' field select 'Part_1' from the 'Parts' label in the 'Object Tree' or from the 'Graphics Viewer'. Then under 'To Object' Select 'Part_2'. This feature will ensure that the software recognises the two parts as one and will be particularly useful later in this tutorial. The final 'OLP Command' window for the 'abb_irb64' Robot should look like that shown below.



Note: As collisions still exist within this Operation, further path points should be added. Specific points for this Operation will be defined at a later stage in this process, however, the steps above should provide the basic knowledge to create and use 'Weld Operations'.

11.4 GRIPPER OPERATIONS: PICK AND PLACE

1. The two parts of this simulation have now been welded together and the next step of the process is to remove the welded parts from the Fixture and place them in a new location where they can be collected for use in a later station. To do this in Process Simulate, a 'Pick and Place' Operation is used.
2. Prior to using a 'Pick and Place' Operation, the clamps of the Fixture must first be opened. As described previously, utilise the 'New Device Control Group Operation' to create a new Operation named 'Clamps_Open_2' which changes the 'Device Control Group' 'Fixture Clamps' from 'Current Pose' to 'OPEN'. Ensure that in doing this the 'Scope' is set to 'StationProcess01'.
3. Now, selecting the 'kr30_3' Robot, create a new 'Pick and Place' Operation by following Operations >> New Operation >> New Pick and Place Operation to display the following dialogue box.



4. Firstly, rename the Operation to: 'Pick_And_Place_Op' then select 'StationProcess01' as the 'Scope'. Within the 'Pick' and 'Place' boxes, the reference points for the Robot should be defined. These points are provided for the purpose of this tutorial and should be input as co-ordinates. This will ensure that the gripper is in the correct position to collect the newly welded parts. Please input the following co-ordinates for this Operation now:

Operation	X	Y	Z	Rx	Ry	Rz
Pick	4379.35	-685.68	994	180	0	180
Place	4270.34	1279.51	702.10	180	0	180

- As with the other Operations created, it is necessary to link it to the previous Operation to ensure it follows the correct sequence of Operations. In addition, at present, the 'Pick and Place' Operation ends at the destination 'Place'. To ensure the Operation runs as desired, please add a location after the 'Place' Location for the Robot position 'HOME'. This is done in "Path Editor" by right-clicking the 'kr30_3' Robot and returning it to the 'Home' position; then right-clicking the 'Place' location select 'Add Current Location'. Rename this location 'Pick_And_Place_End_To_Home'.

At this point in the tutorial, your simulation should be at a stage where clicking 'play' will run through the Operations defined above. These steps should have given you the basic training needed in each of the above Operations, however, there is still a significant number of features which can be utilised in Process Designer and Process Simulate. The following sections of this tutorial are focused on developing a more advanced knowledge of simulation development within Process Simulate specifically collision avoidance, cosmetic additions and the use of the Cyclic Event Evaluator. Using the Simulation built already, additions will be made to take the current factory to a far more developed stage where more features will be explored and the user will be left with a more complete knowledge.

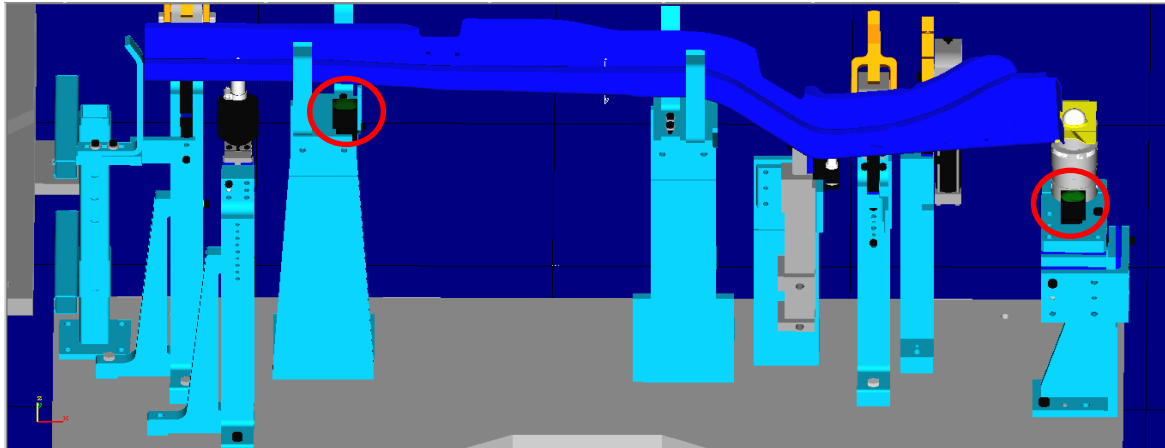
12.0 PLACEMENT OF PROXIMITY SENSORS

- Proximity sensors were added at the initial stages of the factory however were not placed into their correct position. It is now necessary to perform this step as these will be used when logic is applied. Opening the 'Object Tree', expand 'GripperFixture' and right-click each of the proximity sensors in turn. Select 'Placement Manipulator' and enter the following co-ordinates:

X	Y	Z	Rx	Ry	Rz
5172.16	-677.11	666.11	0	0	0
4032.22	-671.65	899.25	0	0	180

The proximity sensors should be located as shown below:

Note: At present these are a graphical representation only and the sensor functionality will be applied at a later stage during the application of logic, Section 17.0.



13.0 ADDING COSMETIC FEATURES

In the tutorial so far, only essential 'Resources' have been utilised. In reality this is not representative of the real factory and so all features should be included. The next step in this tutorial is to import all the necessary parts to complete the look of the factory.

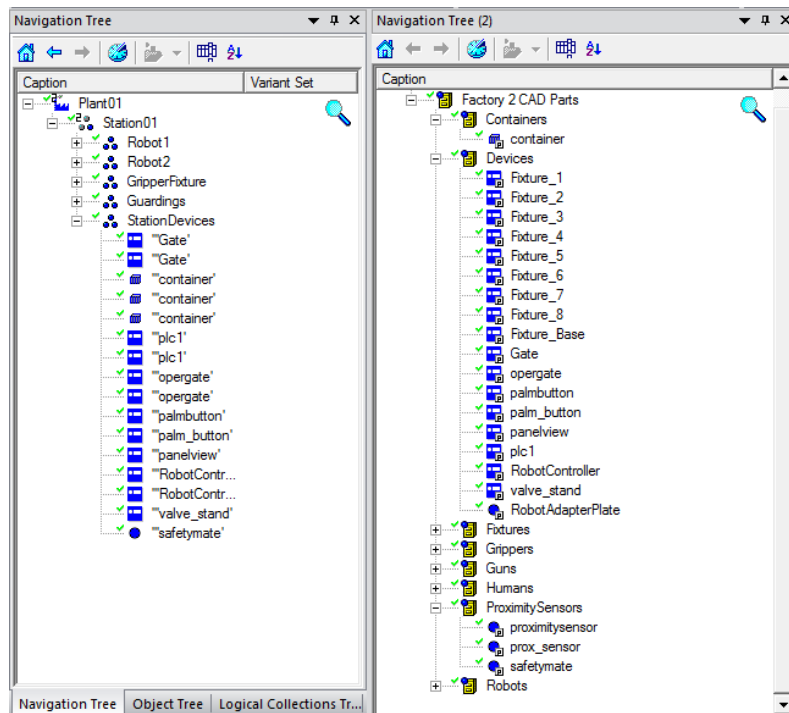
1. Going back to the 'Navigation Tree', open the 'EngineeringResourceLibrary' in a new 'Navigation Tree' alongside the original Tree.
2. Opening 'Station01' it will be possible to drag and drop the necessary cosmetic features into the Plant for use in the simulation. Firstly, create a new 'Compound Resource' within 'Plant01' Renaming this as 'StationDevices', then create another named 'Guardings'. Following this, drag and drop the Resources below into the 'Guardings' Compound Resource. These will be used to create the perimeter fence for the Station.
 - 27 x 'Fence_Post'
 - 13 x 'Fence_Large'
 - 12 x 'Fence_Medium'
 - 5 x 'Fence_Small'

Note: After adding Resources to the 'Plant01' node within the 'Navigation Tree' it may be necessary to re-load the 'RobcadStudy' in order to load the newly added Resources in the Graphic Viewer. Right-click the 'RobcadStudy' and select 'Load in Standard Mode' to allow the Resources to generate.

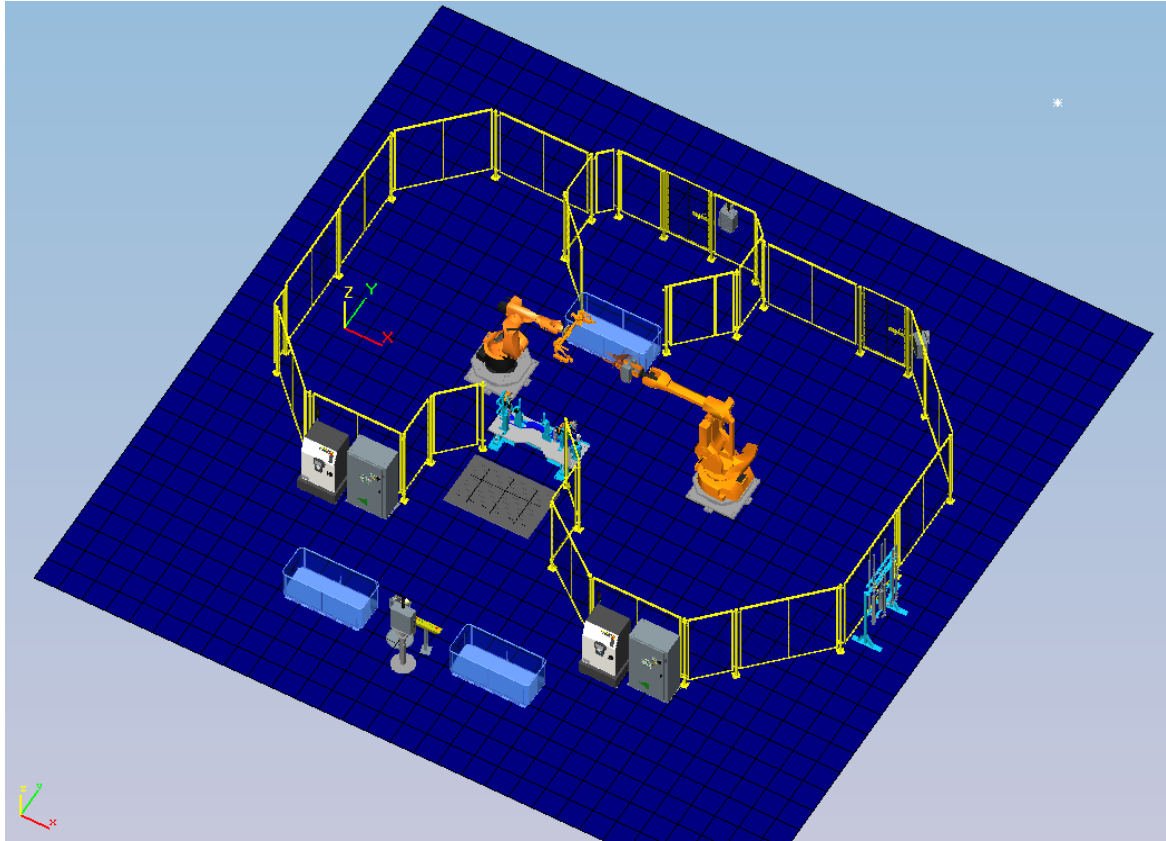
- In the 'Object Tree', right-click each of the Resources within 'Guardings' individually and select 'Placement Manipulator'. In creating the perimeter fence, the co-ordinates for the layout presented in this tutorial are attached as an Excel spreadsheet where each co-ordinate can be input manually. However, the user may take this opportunity to practice manual placement manipulation, re-creating the perimeter fence as close to that shown as possible.

Note: when entering these co-ordinates, it is important to input the Rz co-ordinate first, else the absolute location will be inaccurate.

- With the perimeter fence now in place, other cosmetic components may now be added. In a drag and drop method similar to that described above, please populate the 'StationDevices' CompoundResource with the Resources shown below. Reload the study once again.



- As with the perimeter fencing, these Resources may be placed manually, however, co-ordinates are given in the provided Excel spreadsheet. The figure below should be used as a reference to ensure that the cosmetic features have been added to the Station correctly.



14.0 REFINEMENT OF PATH LOCATIONS

14.1 OBJECT FLOW OPERATION

Since the initial creation of the 'Object Flow Operation' used to place the parts into the Fixture, two containers have been inserted from which the parts are to be sourced; this is to represent the action of a human picking parts from the containers and manually placing them into the Fixture. Given that the location of the buckets does not correspond with the 'Object Flow' starting location however, it is necessary to update the values so that they match. It is also necessary to add additional 'via' locations to ensure that collisions are removed and that the parts move in a controlled motion path.

- Open 'StationProcess01' within the "Path Editor". Selecting 'PART_2_loc_1', right-click and select 'Manipulate Location'. The co-ordinates should be adjusted to match those as shown below.

2. In addition, create a further two 'via' locations by, again, right-clicking 'PART_2_loc_1' and selecting 'Add Location After'. When prompted, set the co-ordinates to match those below.
3. Repeat the above two steps, this time for 'PART_1_loc_1' utilising the values displayed below.
4. Finally, rename all locations to comply with the previous naming convention.

Paths & Locations	Attachment	X	Y	Z	RX	RY	RZ
Clamps_Open							
PART_2_to_Fixture							
PART_2_loc_1		6436.23	-5521.41	140.95	0.00	0.00	180.00
loc		6436.23	-5521.41	1180.68	0.00	0.00	180.00
loc1		4601.79	-3508.20	1180.68	0.00	0.00	180.00
PART_2_loc_2		5119.00	-501.46	1015.00	0.00	0.00	180.00
PART_2_loc_3		5119.00	-501.46	815.59	0.00	0.00	180.00
PART_1_to_Fixture							
PART_1_loc_1		2857.58	-5524.22	246.13	0.00	0.00	180.00
loc		2857.58	-5524.22	1180.68	0.00	0.00	180.00
loc1		4459.72	-3495.07	1180.68	0.00	0.00	180.00
PART_1_loc_2		4448.34	-620.41	1150.00	0.00	0.00	180.00
PART_1_loc_3		4448.34	-620.41	952.60	0.00	0.00	180.00
Clamps_Close							

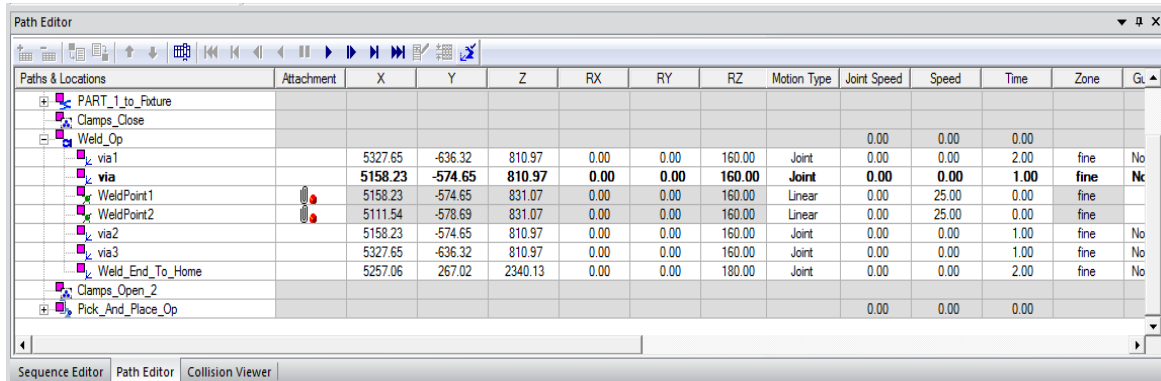
14.2 WELD OPERATION

It is necessary to refine the path of the Robot in completing the 'Weld Operation'. In reality, the Robot is travelling more slowly in the vicinity of parts to minimise the chance of a collision and so this should be reflected within the simulation as closely as possible.

1. Ensure that 'StationProcess01' is the current Operation within the "Path Editor". Reset the simulation then play up until both parts are loaded into the Fixture and pause.
2. Right-click 'WeldPoint1' and select 'Jump Assigned Robot' to show the Robot position at this Weld Point. The next stage is to define the path of the Robot leading up to this Weld Point.
3. Again, right-click 'WeldPoint1' but this time select 'Add Location Before'. Perform this action twice to create two 'via' points prior to 'WeldPoint1' and populate the co-ordinate fields within the "Path Editor" to match those show below.
4. For the Robot to exit on the same path as it entered during the 'Weld Operation', two identical 'via' points must be created after 'WeldPoint2'. This can be achieved by right-clicking the location 'via' and selecting 'Jump Assigned Robot'. Following this, right-click 'WeldPoint2' and select 'Add Current Location'. Repeat this step for the location 'via1' ensuring 'Add Current Location' is selected on the newly created point 'via2'. Alternatively,

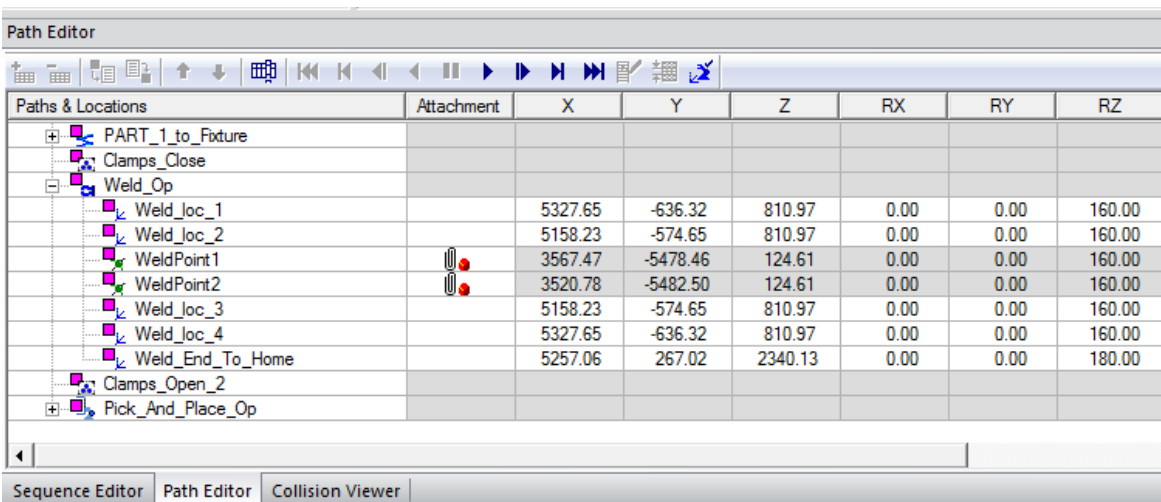
arbitrary 'via' locations can be created and the co-ordinates updated to match those as shown below.

- Finally, adjust the 'Motion Type', 'Speed' and 'Time' to match those shown. This will slow the speed of the Robot motion during the Operation. Linear motion is selected at the Weld Point to increase accuracy of the Robot.



Paths & Locations	Attachment	X	Y	Z	RX	RY	RZ	Motion Type	Joint Speed	Speed	Time	Zone	G-code
PART_1_to_Fixture													
Clamps_Close													
Weld_Op													
via1		5327.65	-636.32	810.97	0.00	0.00	160.00	Joint	0.00	0.00	0.00	fine	No
via		5158.23	-574.65	810.97	0.00	0.00	160.00	Joint	0.00	0.00	1.00	fine	No
WeldPoint1		5158.23	-574.65	831.07	0.00	0.00	160.00	Linear	0.00	25.00	0.00	fine	
WeldPoint2		5111.54	-578.69	831.07	0.00	0.00	160.00	Linear	0.00	25.00	0.00	fine	
via2		5158.23	-574.65	810.97	0.00	0.00	160.00	Joint	0.00	0.00	1.00	fine	No
via3		5327.65	-636.32	810.97	0.00	0.00	160.00	Joint	0.00	0.00	1.00	fine	No
Weld_End_To_Home		5257.06	267.02	2340.13	0.00	0.00	180.00	Joint	0.00	0.00	2.00	fine	No
Clamps_Open_2													
Pick_And_Place_Op									0.00	0.00	0.00		

- Rename the Weld Points to match the names as shown below.



Paths & Locations	Attachment	X	Y	Z	RX	RY	RZ
PART_1_to_Fixture							
Clamps_Close							
Weld_Op							
Weld_loc_1		5327.65	-636.32	810.97	0.00	0.00	160.00
Weld_loc_2		5158.23	-574.65	810.97	0.00	0.00	160.00
WeldPoint1		3567.47	-5478.46	124.61	0.00	0.00	160.00
WeldPoint2		3520.78	-5482.50	124.61	0.00	0.00	160.00
Weld_loc_3		5158.23	-574.65	810.97	0.00	0.00	160.00
Weld_loc_4		5327.65	-636.32	810.97	0.00	0.00	160.00
Weld_End_To_Home		5257.06	267.02	2340.13	0.00	0.00	180.00
Clamps_Open_2							
Pick_And_Place_Op							

14.3 PICK AND PLACE LOCATIONS

The 'Pick and Place' Operation must be adjusted so that the Robot places the final part in the container as, at present, the part is released at an incorrect location.

- Within this Operation, four 'via' points must be created using the procedure stated above. Create one 'via' point before 'Pick', two between 'Pick' and 'Place' and one after 'Place'. In addition, the co-ordinates for the other locations should be checked and updated if

necessary. Please see below for the most up-to-date co-ordinate points including those for the 'via' locations.

Paths & Locations	Attachment	X	Y	Z	RX	RY	RZ
Weld_loc_3		5158.23	-574.65	810.97	0.00	0.00	160.00
Weld_loc_4		5327.65	-636.32	810.97	0.00	0.00	160.00
Weld_End_To_Home		5257.06	267.02	2340.13	0.00	0.00	180.00
Clamps_Open_2							
Pick_And_Place_Op							
via		4379.35	-685.68	1224.62	180.00	0.00	180.00
Pick		4379.35	-685.68	994.00	180.00	0.00	180.00
via1		4379.35	-685.68	1224.62	180.00	0.00	180.00
via2		4374.22	2041.88	1020.14	180.00	0.00	180.00
Place		4374.22	2041.88	317.49	180.00	0.00	180.00
via3		4374.22	2041.88	1020.14	180.00	0.00	180.00
Pick_And_Place_End_To_Home		4572.91	687.24	2122.28	90.00	0.00	90.00

- Following this, rename the locations as shown below. It is also necessary to update the configuration of the 'Pick_And_Place_End_To_Home' location to ensure the Robot returns to its correct 'Home' position. This is done by clicking the 'Configuration' box indicated below. Following this, select the configuration 'J3 – OH –' and click 'Teach'. A 'green' tick should now display in this box.

Paths & Locations	Attachment	X	Y	Z	RX	RY	RZ	Configuration	External Axes	Dynar
Weld_loc_3		5158.23	-574.65	810.97	0.00	0.00	160.00		0 out of 1	
Weld_loc_4		5327.65	-636.32	810.97	0.00	0.00	160.00		0 out of 1	
Weld_End_To_Home		5257.06	267.02	2340.13	0.00	0.00	180.00		0 out of 1	
Clamps_Open_2										
Pick_And_Place_Op										
Pick_And_Place_loc_1		4379.35	-685.68	1224.62	180.00	0.00	180.00			
Pick		4379.35	-685.68	994.00	180.00	0.00	180.00			
Pick_And_Place_loc_2		4379.35	-685.68	1224.62	180.00	0.00	180.00			
Pick_And_Place_loc_3		4374.22	2041.88	1020.14	180.00	0.00	180.00			
Place		4374.22	2041.88	317.49	180.00	0.00	180.00			
Pick_And_Place_loc_1Pick_And...		4374.22	2041.88	1020.14	180.00	0.00	180.00			
Pick_And_Place_End_To_Home		4572.91	687.24	2122.28	90.00	0.00	90.00	✓		

You have now completed the second stage of the process – Creating the Simulation on Process Simulate. Next you will move onto creating the associated control program; this will predominately use the Cyclic Event Evaluator (CEE) within Process Simulate.

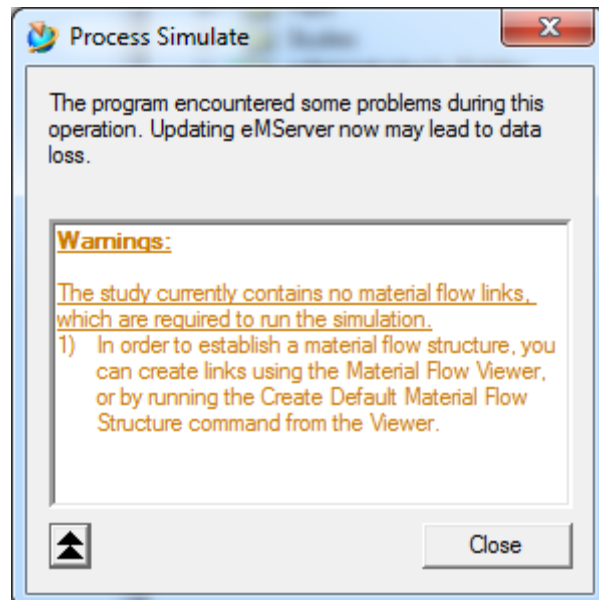
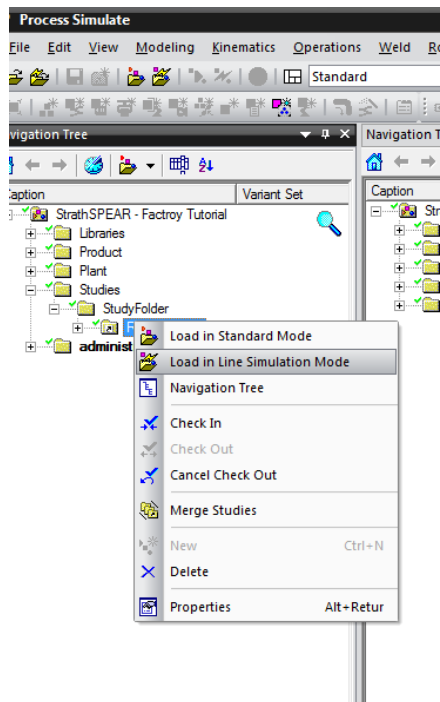
CHAPTER 3 – CYCLIC EVENT EVALUATOR


15. DEFINITION OF MATERIAL FLOW

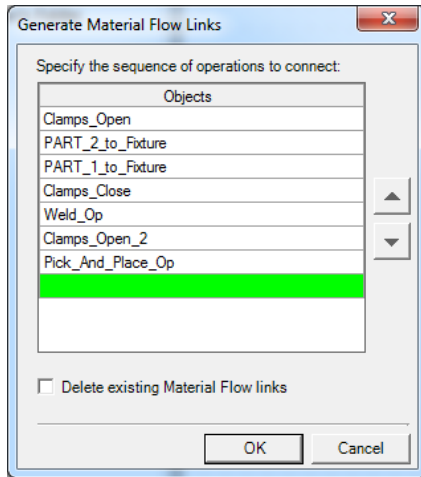
Material flow is a function which defines the introduction and consumption of material parts used within the simulation. This function provides a visual representation of the existence and use of parts within a manufacturing facility during Cyclic Operations.

Note: When working in 'Line Simulation' mode (used for Cyclic Operations) parts are only viewed in the simulation as appearances and are only visually represented in the 'Graphics Viewer' during their consumption by an Operation.

1. Right click the 'Robcad Study' on the 'Navigation Tree' and select 'Load in Line Simulation Mode' as displayed on the left below.



2. Ignore the warning that appears on the screen by clicking close (see above on the right) as you will now be defining the material flow structure.
3. Open 'Operation Tree' and expand all levels.
4. On the main toolbar select View >> Viewers >> Material Flow Viewer.
5. Within the 'Material Flow Viewer', click the 'Generate Material Flow Links'  icon.
6. Now click the green 'Objects' taskbar and this will allow you to add Operations as objects.
7. Using the 'Operation Tree' you must now add Operations. The Operations that must be added are detailed in the following window screenshot:



- Click OK to close the 'Generate Material Flow Links' window.

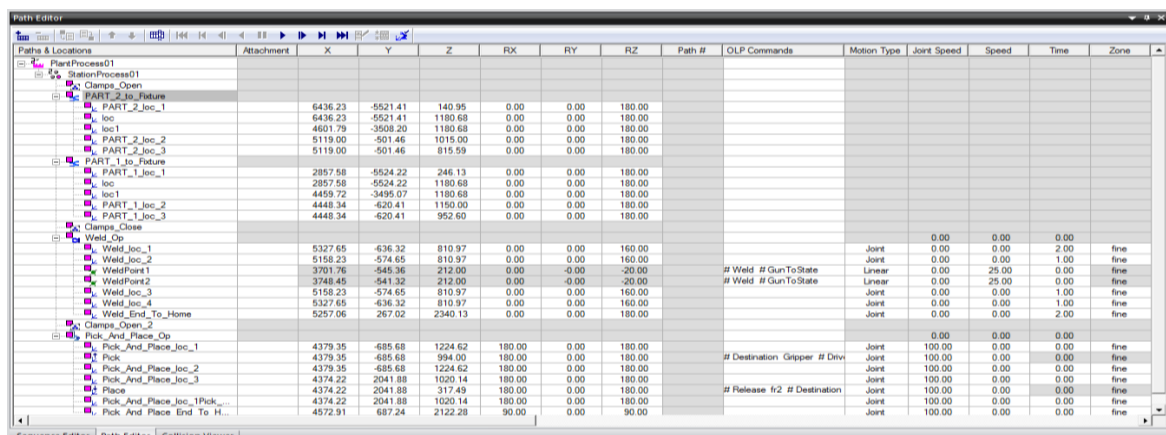
Note: The material flow for the project has now been defined and can be observed in the 'Material Flow Viewer'.

16.0 DEFINITION OF JOINT VALUE SENSORS

It is now necessary to introduce 'Joint Value Sensors' for each of the Robots. These sensors will allow specific Robot poses to be defined for use later in the simulation.

16.1 ABB_IRB64 ROBOT JOINT VALUE SENSORS

- Within the 'Operations Tree' drag the 'PlantProcess01' into the 'Path Editor' and expand all levels. The 'Path Editor' should now look like the following:

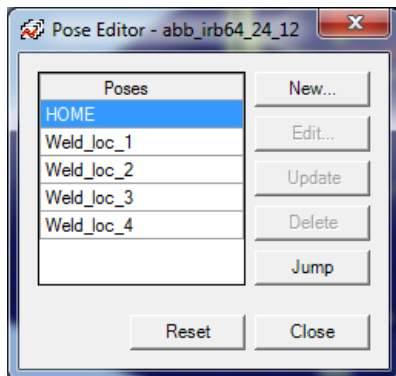


Attachment	X	Y	Z	RX	RY	RZ	Path #	OLP Commands	Motion Type	Joint Speed	Speed	Time	Zone
PlantProcess01													
StationProcess01													
Clamps_Open													
PART_2_to_Fixture													
loc	6436.23	-5521.41	140.95	0.00	0.00	180.00							
loc1	6436.23	-5521.41	1180.68	0.00	0.00	180.00							
loc1	4601.79	-3508.20	1180.68	0.00	0.00	180.00							
PART_2_Joc_2	5119.00	-501.46	1015.00	0.00	0.00	180.00							
PART_2_Joc_3	5119.00	-501.46	815.59	0.00	0.00	180.00							
PART_1_to_Fixture													
PART_1_Joc_1	2857.58	-5524.22	248.13	0.00	0.00	180.00							
loc	2857.58	-5524.22	1180.68	0.00	0.00	180.00							
loc1	4459.72	-3495.07	1180.68	0.00	0.00	180.00							
PART_1_Joc_2	4448.34	-620.41	1150.00	0.00	0.00	180.00							
PART_1_Joc_3	4448.34	-620.41	952.60	0.00	0.00	180.00							
Clamps_Close													
Weld_Op													
Weld_Joc_1	5327.65	-636.32	810.97	0.00	0.00	160.00			Joint	0.00	0.00	0.00	
Weld_Joc_2	5158.23	-574.65	810.97	0.00	0.00	160.00			Joint	0.00	0.00	2.00	fine
WeldPoint1	3701.76	-545.36	212.00	0.00	-0.00	-20.00		# Weld # GunToState	Linear	0.00	25.00	0.00	fine
WeldPoint2	3748.45	-541.32	212.00	0.00	-0.00	-20.00		# Weld # GunToState	Linear	0.00	25.00	0.00	fine
Weld_Joc_3	5158.23	-574.65	810.97	0.00	0.00	160.00			Joint	0.00	0.00	1.00	fine
Weld_Joc_4	5327.65	-636.32	810.97	0.00	0.00	160.00			Joint	0.00	0.00	1.00	fine
Weld_End_To_Home	5257.06	267.02	2340.13	0.00	0.00	180.00			Joint	0.00	0.00	2.00	fine
Pick_And_Place_Op													
Pick_And_Place_Joc_1	4379.35	-685.68	1224.62	180.00	0.00	180.00			Joint	100.00	0.00	0.00	fine
Pick	4379.35	-685.68	994.00	180.00	0.00	180.00		# Destination Gripper # Drive	Joint	100.00	0.00	0.00	fine
Pick_And_Place_Joc_2	4379.35	-685.68	1224.62	180.00	0.00	180.00			Joint	100.00	0.00	0.00	fine
Pick_And_Place_Joc_3	4374.22	2041.88	1020.14	180.00	0.00	180.00			Joint	100.00	0.00	0.00	fine
Place	4374.22	2041.88	317.49	180.00	0.00	180.00		# Release fr2 # Destination	Joint	100.00	0.00	0.00	fine
Pick_And_Place_Joc_1Pick...	4374.22	2041.88	1020.14	180.00	0.00	180.00			Joint	100.00	0.00	0.00	fine
Pick_And_Place_End_To H...	4572.91	687.24	2122.28	90.00	0.00	90.00			Joint	100.00	0.00	0.00	fine

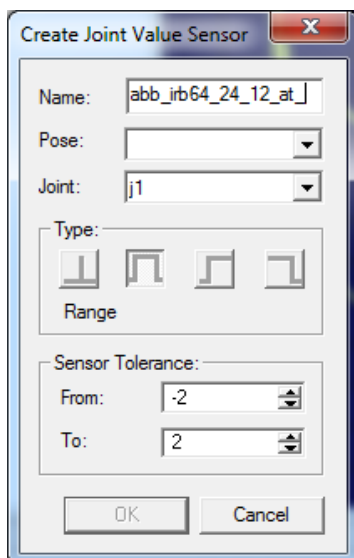
2. In the 'Graphics Viewer' click the 'abb_irb64' Robot (it will turn green) and within the 'Weld_Op' in the 'Path Editor' right click 'Weld_loc_1' and click 'Jump Assigned Robot'. This will move the Robot to the Weld_loc_1 position.
3. Right click the 'abb_irb64' Robot in the 'Graphics Viewer' and select 'Mark Pose'.
4. Right click the 'abb_irb64' Robot again and select Pose Editor >> Pos1 >> Edit and change the name to 'Weld_loc_1'.
5. Repeat steps 2-4 for Weld_loc_ numbers 1 to 4.

Note: You will be unable to create poses for WeldPoint1 and WeldPoint2. Do not worry as these are not required.

Upon completion of Step 5, the 'Pose Editor' window should look like the following:



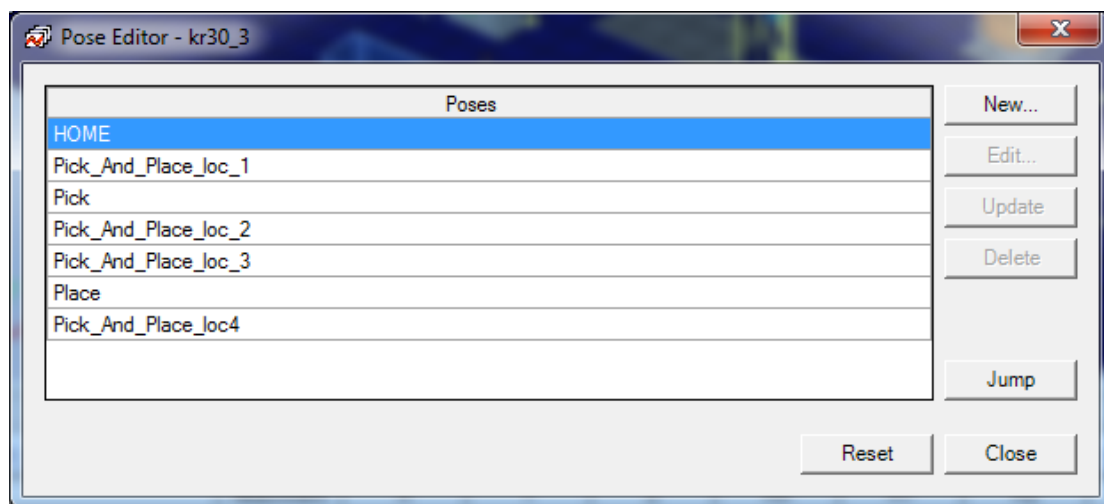
6. Select the 'abb_irb64' Robot in the 'Graphics Viewer'. Then, in the main toolbar click CEE >> Sensors >> Create Joint Value Sensors. The following window will appear:



7. Select the first pose in the 'Pose' dropdown menu (this will be HOME) and change the name to 'abb_irb64_24_12_at_HOME'.
8. Repeat step 7 for all other poses in the 'Pose' dropdown menu, changing the names to the corresponding names of the poses.

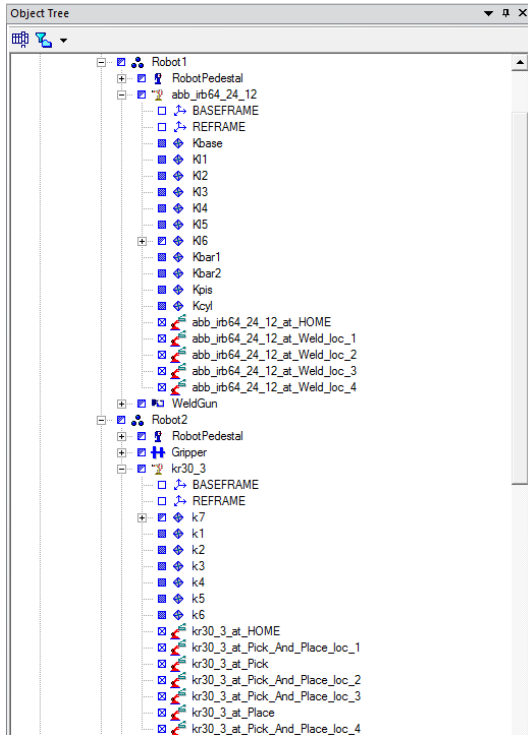
16.2 KR30_3 ROBOT JOINT VALUE SENSORS

1. Right click the 'kr30_3' Robot in the 'Graphics Viewer' (it will turn green).
2. Now within the 'Pick_And_Place_Op' in the "Path Editor", right click 'Pick_And_Place_loc_1' and click 'Jump Assigned Robot'. This will move the Robot to the Pick_And_Place_loc_1 position.
3. Right click the 'kr30_3' Robot in the 'Graphics Viewer' and select 'Mark Pose'.
4. Right click 'kr30_3' Robot again and select Pose Editor >> Pos1 >> Edit and change the name to 'Pick_And_Place_loc_1'.
5. Repeat steps 2-4 for all positions in the 'Pick_And_Place_Op' apart from 'Pick_And_Place_End_To_Home'.
6. Upon completion of step 5 the pose editor should resemble the following:



7. Click the 'kr30_3' Robot in the 'Graphics Viewer'.
8. In the main toolbar select CEE >> Sensors >> Create Joint Value Sensors.
9. Select the first pose in the 'Pose' dropdown menu (HOME) change the name to 'kr30_3_at_HOME'.
10. Repeat step 9 for all other poses in the 'Pose' dropdown menu, changing the names to the corresponding names of the poses.

Joint Value Sensors have now been defined for both of the Robots. These poses set by each Joint Value Sensors should be shown in the 'Object Tree' as shown below in the following screenshot:

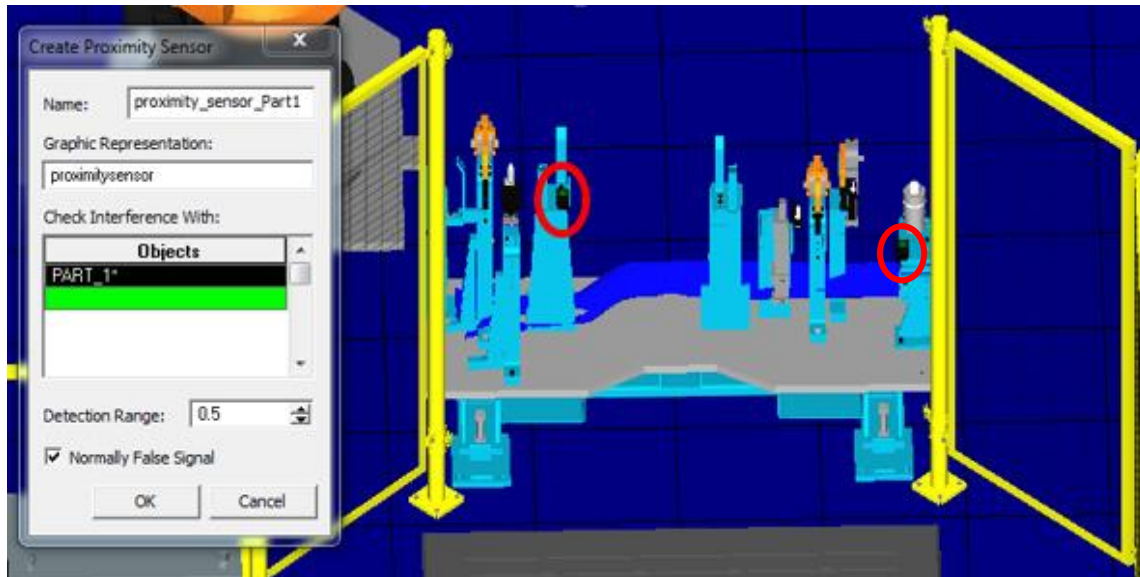


Note: Although all poses have been defined using Joint Value Sensors. Only the HOME poses for each Robot will be used in the logic. However it must be noted that in other projects additional poses may become applicable for the logic.

17.0 PROXIMITY SENSORS

Proximity Sensors are used for part and human detection within the simulation and must be represented by individual CAD models within the simulation. For the purpose of this project they will be used to identify the presence of both 'PART_1*' and 'PART_2*' within the clamp Fixture.

1. CEE >> Create Proximity Sensor.
2. Within the 'Name' box type: 'proximity_sensor_Part1'.
3. Select the 'Graphic Representation' box and then click the proximity sensor on the left-hand side of the Fixture as shown by the red circle below.




4. Open 'Operation Tree', right click LineOperation >> Generate Appearances.
5. Select the 'Check Inference With' objects box and then within the 'Object Tree', select 'PART_1*'. This will add 'PART_1*' to the Objects window.
6. Set the 'Detection Range' to 1000 and ensure that 'Normally False Signal' is ticked/checked. Close the window by clicking ok. The sensor will now appear in the 'Object Tree' under Plant01 node.
7. Repeat steps 1-6 for the second proximity sensor (Shown above in the right-hand circle). Please note that the 'Check Interference With' object for the second proximity sensor will be 'PART_2*'.

18.0 SIGNAL GENERATION

For logic to be added to the simulation, Signal Generation is key. Input and Output signals must be created for all Robots and Devices. With the use of logical expressions these signals are then used to either permit or restrict the start of Operations within the simulation.

18.1 CREATION OF ROBOT SIGNALS

1. In the 'Graphics Viewer' select the 'abb_irb64' Robot and within the main toolbar at the top of the screen select Robotics >> Robot Signals. This will open the Robot signals window.
2. Now click the 'Create Default Signals'  icon. The following will be displayed:

Robot Signals - " abb_irb64_24_12 "

PLC Signal Name	Robot Signal Name	I/O	Signal Function	HW T...	Address	External Connection	Comments
abb_irb64_24_12_at_Weld_loc_1	Weld_loc_1	I	Pose Signal	BOOL	No Address		
abb_irb64_24_12_at_HOME	HOME	I	Pose Signal	BOOL	No Address		
abb_irb64_24_12_at_Weld_loc_4	Weld_loc_4	I	Pose Signal	BOOL	No Address		
abb_irb64_24_12_at_Weld_loc_3	Weld_loc_3	I	Pose Signal	BOOL	No Address		
abb_irb64_24_12_at_Weld_loc_2	Weld_loc_2	I	Pose Signal	BOOL	No Address		
abb_irb64_24_12_startProgram	startProgram	Q	Starting Program	BOOL	No Address		
abb_irb64_24_12_programNumber	programNumber	Q	Program Number	BYTE	No Address		
abb_irb64_24_12_emergencyStop	emergencyStop	Q	Program Emergency Stop	BOOL	No Address		
abb_irb64_24_12_programPause	programPause	Q	Program Pause	BOOL	No Address		
abb_irb64_24_12_programEnded	programEnded	I	Ending Program	BOOL	No Address		
abb_irb64_24_12_mirrorProgramNumber	mirrorProgramNumber	I	Mirror Program Number	BYTE	No Address		
abb_irb64_24_12_errorProgramNumber	errorProgramNumber	I	Error Program Number	BOOL	No Address		
abb_irb64_24_12_robotReady	robotReady	I	Robot Ready	BOOL	No Address		

OK Cancel Apply

3. Click 'Apply' and 'OK' to close the window.
4. Select the 'abb_irb64' Robot and click CEE >> Signal Generation >> Create Robot Start Signals on the main toolbar.
5. Repeat Steps 1 and 2 for the 'kr30_3' Robot. The following signals will be displayed:

Robot Signals - " kr30_3 "

PLC Signal Name	Robot Signal Name	I/O	Signal Function	HW T...	Address	External Connection	Comments
kr30_3_at_Pick_And_Plac...	Pick_And_Place_loc_1	I	Pose Signal	BOOL	No Address		
kr30_3_at_Pick_And_Plac...	Pick_And_Place_loc_3	I	Pose Signal	BOOL	No Address		
kr30_3_at_Pick_And_Plac...	Pick_And_Place_loc_2	I	Pose Signal	BOOL	No Address		
kr30_3_at_Pick	Pick	I	Pose Signal	BOOL	No Address		
kr30_3_at_Pick_And_Plac...	Pick_And_Place_loc4	I	Pose Signal	BOOL	No Address		
kr30_3_at_HOME	HOME	I	Pose Signal	BOOL	No Address		
kr30_3_at_Place	Place	I	Pose Signal	BOOL	No Address		
kr30_3_startProgram	startProgram	Q	Starting Program	BOOL	No Address		
kr30_3_programNumber	programNumber	Q	Program Number	BYTE	No Address		
kr30_3_emergencyStop	emergencyStop	Q	Program Emergency Stop	BOOL	No Address		
kr30_3_programPause	programPause	Q	Program Pause	BOOL	No Address		
kr30_3_programEnded	programEnded	I	Ending Program	BOOL	No Address		
kr30_3_mirrorProgramNum...	mirrorProgramNumber	I	Mirror Program Number	BYTE	No Address		
kr30_3_errorProgramNumber	errorProgramNumber	I	Error Program Number	BOOL	No Address		
kr30_3_robotReady	robotReady	I	Robot Ready	BOOL	No Address		

OK Cancel Apply

6. Select the 'kr30_3' Robot, click CEE >> Signal Generation >> Create Robot Start Signals.

Note: At this stage, all signals for the 'abb_irb64' Robot and 'kr30_3' Robot have been created.

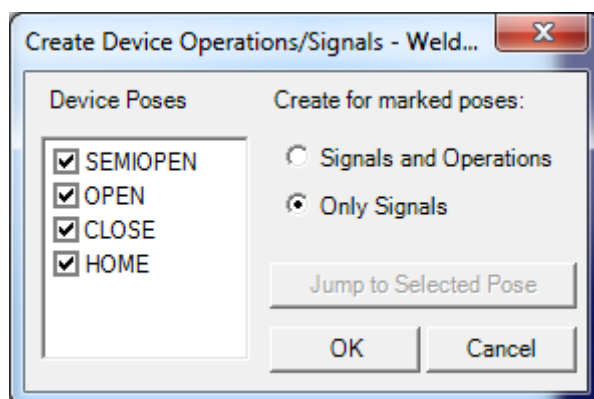
In this section signals have been created for the poses (defined in the Section 15.2), Robot Operation start signals and default Robot signals.

For the purpose of this tutorial not all of these default Robot signals will be used. The signals of importance are as follows:-

- 1) 'abb_irb64_at_HOME' and 'kr30_3_at_HOME' – These signals will be used to return the Robot to its home position*
- 2) 'abb_irb64_startProgram' and 'kr30_3_startProgram' - These will be used to signal the initiation of Robot Programs*
- 3) 'abb_irb64_programNumber' and 'kr30_3_programNumber' – Used to define the specific Robot Program to be used by the 'abb_irb64_startProgram' and 'kr30_3_startProgram' signals respectively.*
- 4) 'abb_irb64_programPause' and 'kr30_3_programPause' – Will be used in the definition of safety features later in the simulation.*

18.2 DEVICE SIGNALS

1. Select the 'WeldGun' in the 'Graphics Viewer' (this device is attached to the 'abb_irb64' Robot).
2. Select CEE >> Signal Generation >> Create Device Operations / Signals and complete the window as shown below:

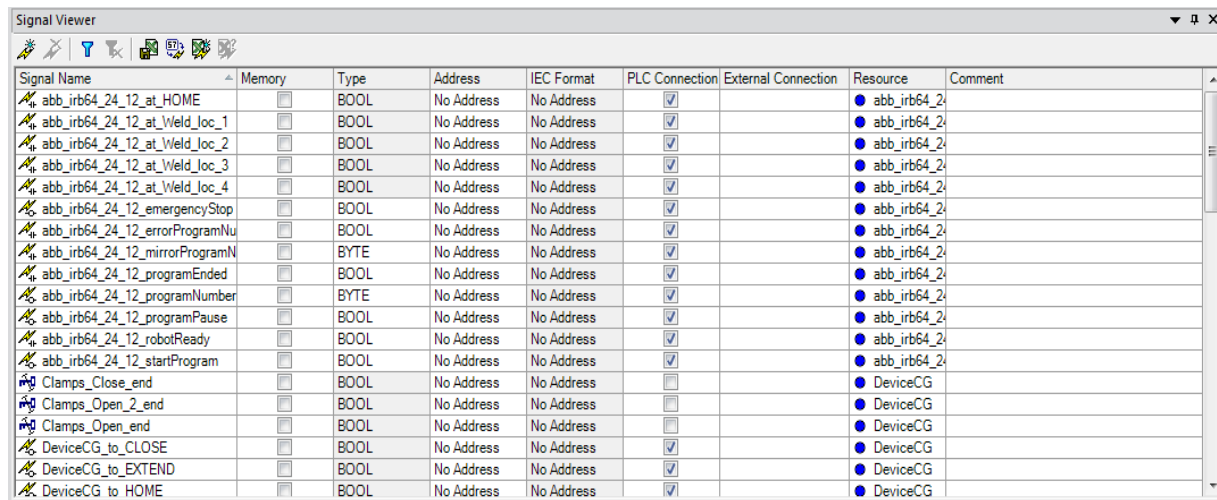


Note: The poses of the device can be observed in the ‘Graphics Viewer’ by ticking an individual pose and clicking the ‘Jump to Selected Pose’ tab. This is a good way to verify that the poses are correct.

3. Select the ‘Gripper’ in the ‘Graphics Viewer’ (this device is attached to the kr30_3 Robot).
4. Repeat step 2 and within the ‘Create Device Operations / Signals’ window, ensure that all ‘Device Poses’ are checked and that the ‘Only Signals’ option is selected.

18.3 USING THE SIGNAL VIEWER

The created signals are displayed in the ‘Signal Viewer’ which is accessed via the main toolbar, selecting View >> Viewers >> Signal Viewer. A portion of the ‘Signal Viewer’ can be seen below:




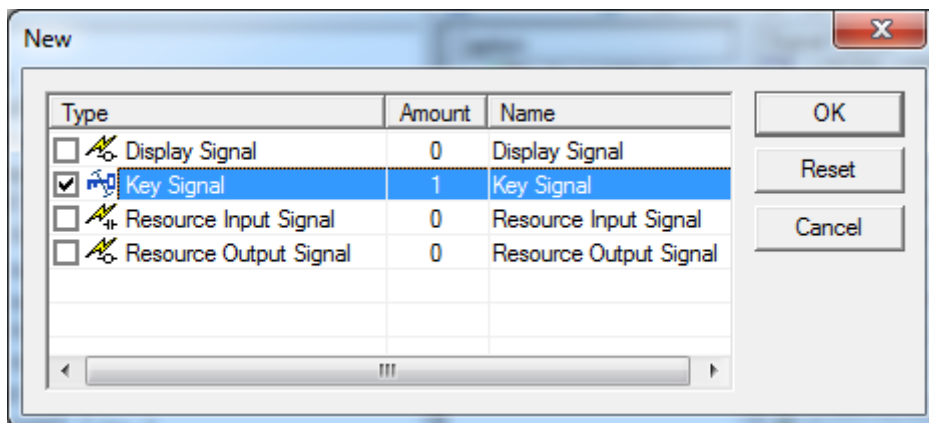
Signal Name	Memory	Type	Address	IEC Format	PLC Connection	External Connection	Resource	Comment
abb_irb64_24_12_at_HOME	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
abb_irb64_24_12_at_Weld_loc_1	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
abb_irb64_24_12_at_Weld_loc_2	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
abb_irb64_24_12_at_Weld_loc_3	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
abb_irb64_24_12_at_Weld_loc_4	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
abb_irb64_24_12_emergencyStop	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
abb_irb64_24_12_errorProgramNu	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
abb_irb64_24_12_mirrorProgramN	<input type="checkbox"/>	BYTE	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
abb_irb64_24_12_programEnded	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
abb_irb64_24_12_programNumber	<input type="checkbox"/>	BYTE	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
abb_irb64_24_12_programPause	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
abb_irb64_24_12_robotReady	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
abb_irb64_24_12_startProgram	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● abb_irb64_24	
Clamps_Close_end	<input type="checkbox"/>	BOOL	No Address	No Address	<input type="checkbox"/>		● DeviceCG	
Clamps_Open_2_end	<input type="checkbox"/>	BOOL	No Address	No Address	<input type="checkbox"/>		● DeviceCG	
Clamps_Open_end	<input type="checkbox"/>	BOOL	No Address	No Address	<input type="checkbox"/>		● DeviceCG	
DeviceCG_to_CLOSE	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● DeviceCG	
DeviceCG_to_EXTEND	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● DeviceCG	
DeviceCG to HOME	<input type="checkbox"/>	BOOL	No Address	No Address	<input checked="" type="checkbox"/>		● DeviceCG	

Note: Within the ‘Signal Viewer’, click ‘Signal Name’ – this organises the signals into alphabetical order which makes it easier when selecting signals for later use.

18.4 CREATING THE START CYCLE SIGNAL

It is necessary to create a signal that can be used to initiate the simulation. This signal takes the form of a ‘Key Signal’.

1. Within the 'Signal Viewer' select the 'Create New Signal'  button.
2. In the dialogue box that appears, check the 'Key Signal' box as shown below:




Click 'OK' to close this dialogue box.

3. A 'Key Signal' will be created. Find the 'Key Signal' within the 'Signal Viewer' by scrolling through the list until the highlighted 'Key Signal' is found. Double-click the signal and rename: 'START CYCLE'. This signal will be used later in the process.

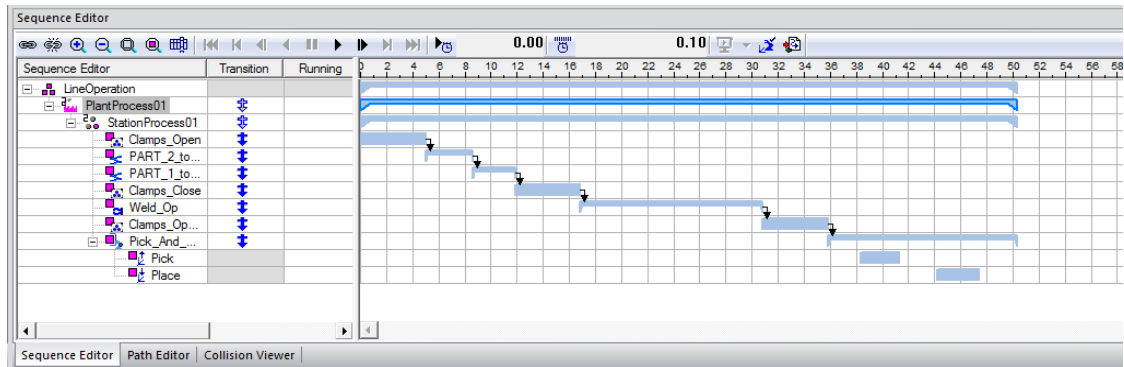
19.0 INTRODUCING LOGIC TO THE SIMULATION


19.1 TRANSITIONS

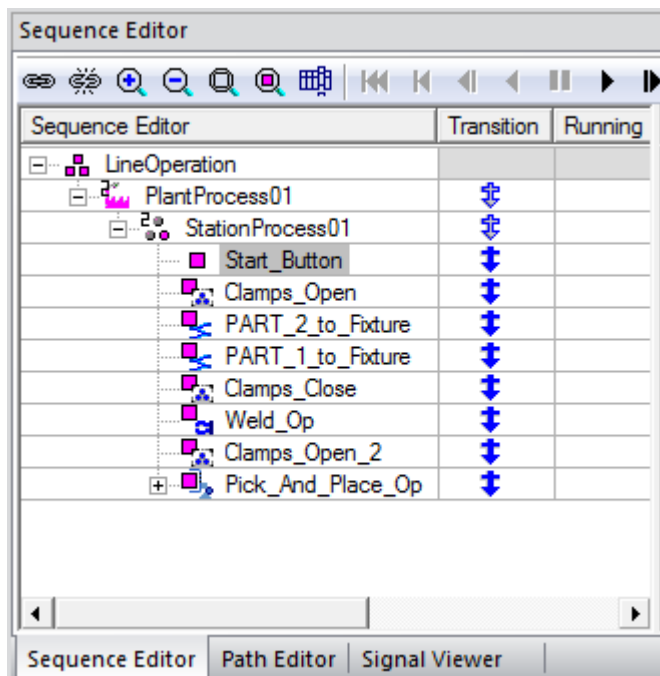
In a time-based simulation, the Sequence of Operations determines the order in which the Operations are conducted whereas in an event based simulation, the logic applied controls the order. The method used to initiate the start of an Operation is the evaluation of the 'Transition Conditions'. 'Transitions' are the link between the Operations and are represented by black

arrows, as shown: . 'Transitions' will be created for the full simulation. This process is detailed as follows:


1. Open the 'Operation Tree', right click 'LineOperation' and select 'Set Current Operation', this will bring the 'LineOperation' into the 'Sequence Editor'. Expand all levels. The resulting 'Sequence Editor' will be displayed:




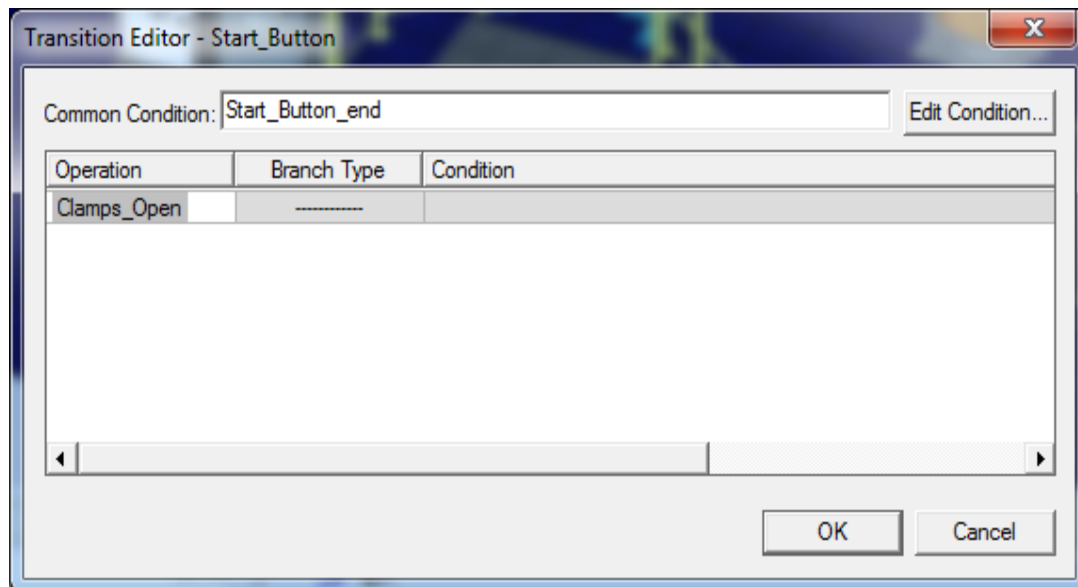
2. Within the 'Sequence Editor', click the  button to add additional columns. For this tutorial you will require the following columns: 'Transition', 'Running', 'Input' and 'Output'.
3. It is now necessary to create a 'Non-Sim Operation'. This will allow a 'Transition' to be created and hence allow the addition of required logic. Firstly, select 'StationProcess01' within the 'Operation Tree'. Now navigate to Operations >> New Operation >> New Non-Sim Operation from the main menu toolbar. A dialogue window will appear; enter the 'Name' as 'Start_Button' and select 'OK'. Finally, click and drag the newly created 'Start_Button' Operation so it becomes the first Operation within 'StationProcess01' as shown below:



4. It is required to create a 'link' from the 'Start_Button' Operation to the 'Clamps_Open' Operation in order to enter logic during the 'Transition'. Select both the 'Start_Button' and 'Clamps_Open' by holding down the 'Ctrl' key. It is important that the 'Start_Button'

Operation is selected before the 'Clamps_Open' Operation .Now click the  icon to create the link.

- Having established a 'link' between all of the Operations, logic can now be applied. When the simulation satisfies the logical commands entered at 'Transition' points, the following Operation in the sequence will commence. This is done by double-clicking the blue 'Transition' icon  beside the 'Start_Button' Operation. A dialogue box will display as shown:



- Click the 'Edit Condition...' button. This will allow you to edit the logic within the 'Common Condition' that has to be satisfied for the simulation to advance to the following Operation in the sequence.

Note: When you start typing to add expressions to the Common Condition a drop down menu will appear containing all signals that you can use.

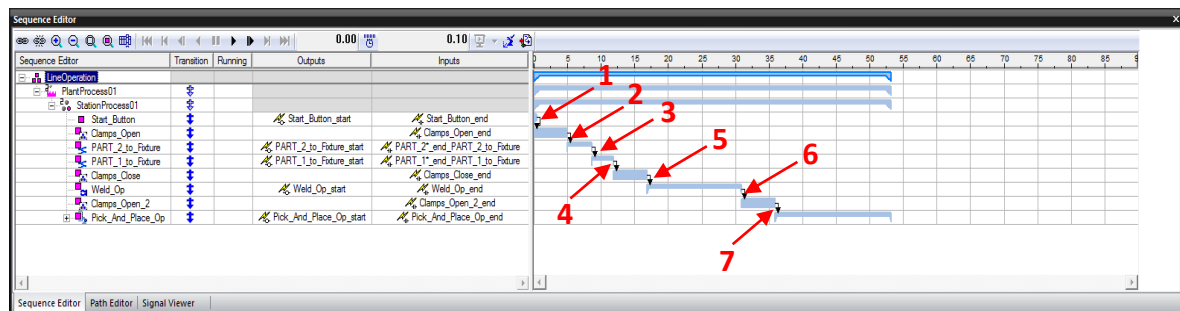
- Enter the 'Common Condition' as:
 'kr30_3_at_HOME **AND** abb_irb64_24_12_at_HOME **AND NOT** proximity_sensor_Part1 **AND NOT** proximity_sensor_Part2'

Note: The 'AND' command must be entered in capital letters.

Note: Logical expressions are input into the 'Transitions' using 'AND', 'AND NOT' and 'OR' statements. It is important to understand the meaning of these commands:

- **AND:** requires the signals either side of the 'AND' statement to be true. An example is: Signal 1 AND Signal 2. This states that Signal 1 and Signal 2 must be true to progress to the next stage of the simulation. It should be noted that a logical expression can contain multiple 'AND' statements and is not restricted to just an expression between two signals.
- **AND NOT:** is the opposite of the 'AND' statement. An example is: Signal 1 AND NOT Signal 2. This states that Signal 1 must be true and Signal 2 must be false to progress to the next stage of the simulation.
- **OR:** allows the simulation to progress with a range of options. An example is: Signal 1 OR Signal 2. Either Signal 1 or Signal 2 must be true to progress to the next stage of the simulation.

8. The 'Transitions' have been labelled from 1-7 in the following diagram:




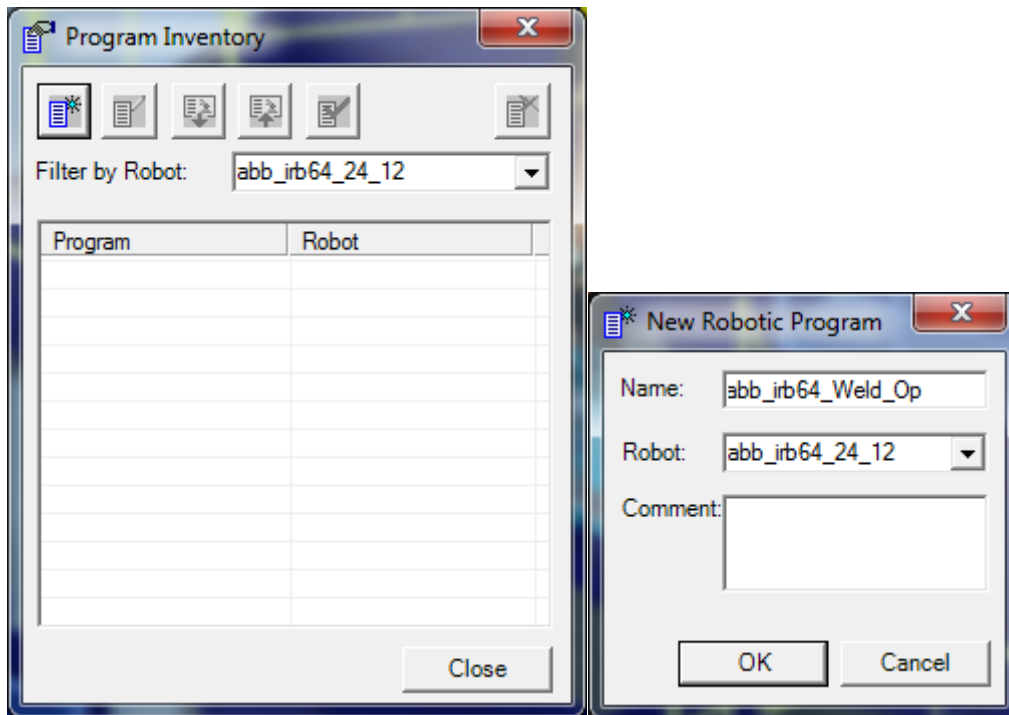
9. With reference to the above diagram input the following common conditions for each transition as per steps 4-7.



Transition #	Common Condition
1	Start_Button_end AND kr30_3_at_HOME AND abb_irb64_24_12_at_HOME AND NOT proximity_sensor_Part1 AND NOT proximity_sensor_Part2
2	Clamps_Open_end AND (NOT proximity_sensor_Part1) AND (NOT proximity_sensor_Part2) AND abb_irb64_24_12_at_HOME
3	PART_2*_end_PART_2_to_Fixture AND proximity_sensor_Part2 AND abb_irb64_24_12_at_HOME
4	PART_1*_end_PART_1_to_Fixture AND proximity_sensor_Part1 AND proximity_sensor_Part2
5	Clamps_Close_end AND proximity_sensor_Part1 AND proximity_sensor_Part2 AND abb_irb64_24_12_at_HOME
6	Weld_Op_end AND proximity_sensor_Part1 AND proximity_sensor_Part2
7	Clamps_Open_2_end AND proximity_sensor_Part1 AND proximity_sensor_Part2

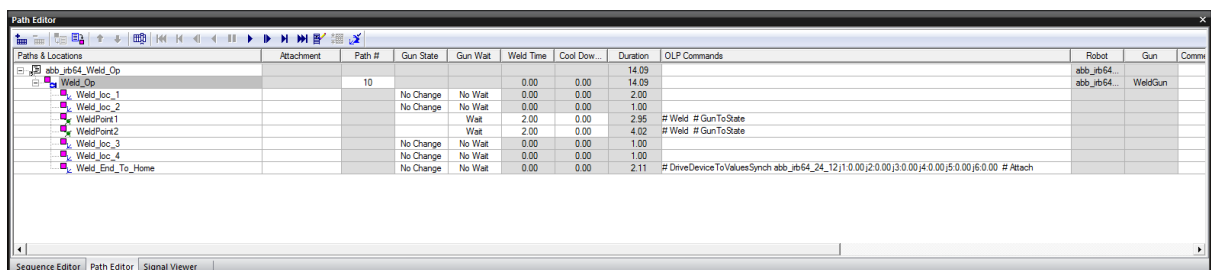
19.2 ROBOT PROGRAMS

A real Robot contains the information for several tasks to be executed. As such, a Robot Program is constructed on Process Simulate to collate the Operations.

1. Click the 'abb_irb64' Robot within the 'Graphics Viewer' and then on the main toolbar Robotics >> Robot Program >> Robot Program Inventory. Within the dialogue box that appears, shown below, click the 'Create New Program' button . In this window 'Name' the New Robotic Program 'abb_irb64_Weld_Op' as highlighted below. Select 'OK' to close the window.



2. Now within the 'Program Inventory' window, select the program that has newly been created ('abb_irb64_Weld_Op') and click the 'Set as Default Program' icon . This will result in the 'abb_irb64_Weld_Op' being shown in bold.
3. Repeat steps 1 and 2 for the 'kr30_3' Robot, naming the Robot Program 'kr30_3_Pick_And_Place_Op'.
4. Select the 'abb_irb64_Weld_Op' Robotic Program and click 'Open in Program Editor'. The Robot Program will be shown in the "Path Editor".
5. Now select the 'Weld_Op' Operation within the 'Operation Tree' and by dragging into the 'add_irb64_Weld_Op' or by using the 'Add Operations to Editor' icon  within 'Path Editor'. Under the column 'Path #' for the 'Weld_Op' enter '10' as displayed in the figure below.




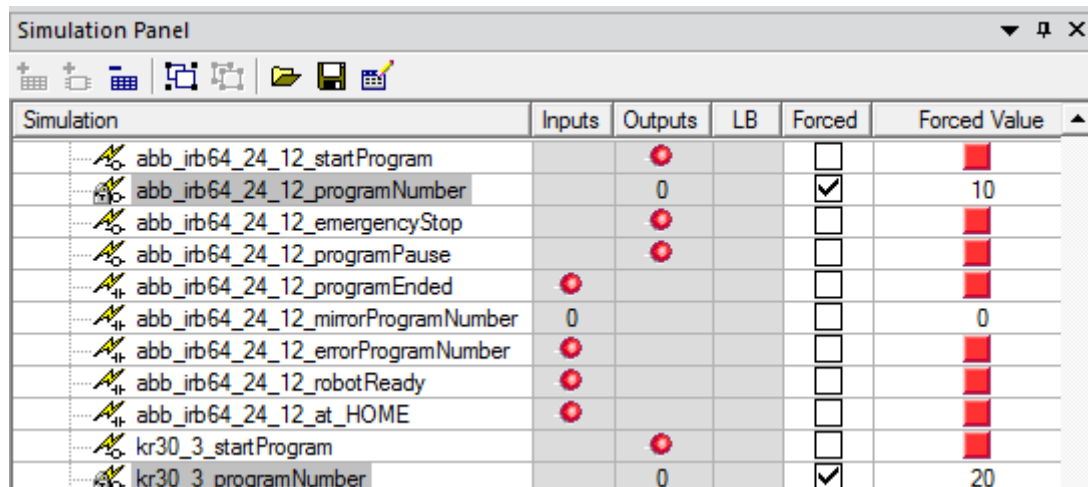
Paths & Locations	Attachment	Path #	Gun State	Gun Wait	Weld Time	Cool Dow...	Duration	OLP Commands	Robot	Gun	Comm
abb_irb64_Weld_Op							14.09		abb_irb64		
Weld_Op		10			0.00	0.00	14.09		abb_irb64	WeldGun	
Weld_Isc_1			No Change	No Wait	0.00	0.00	2.00				
Weld_Isc_2			No Change	No Wait	0.00	0.00	1.00				
WeldPost1				Wait	2.00	0.00	2.95	# Weld # GunToState			
WeldPost2				Wait	2.00	0.00	4.02	# Weld # GunToState			
Weld_Isc_3			No Change	No Wait	0.00	0.00	1.00				
Weld_Isc_4			No Change	No Wait	0.00	0.00	1.00				
Weld_End_To_Home			No Change	No Wait	0.00	0.00	2.11	# DriveDevice ToValuesSynch abb_irb64_24_12;1:0.00;2:0.00;3:0.00;4:0.00;5:0.00;6:0.00 # Attach			




























6. Open Robotics >> Robot Program >> Robot Program Inventory and select the 'kr30_3_Pick_And_Place_Op' and click on 'Open in Program Editor'.

- In the 'Operation Tree' select 'Pick_And_Place_Op' and add to 'Path Editor'. Under the 'Path #' column for the 'Pick_And_Place_Op' enter '20'.

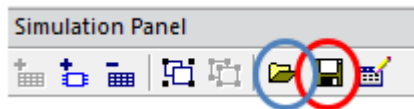
Note: The Path numbers (#) 10 and 20 are arbitrary values used for clarity to distinguish between the 'Weld_Op' and 'Pick_And_Place_Op'. They are not representative of the number of paths within the project.

- Open the following windows: View >> Viewers >> Simulation Panel and View >> Viewers >> Signal Viewer.
- Add all signals from the 'Signal Viewer' to the 'Simulation Panel'. This is conducted by highlighting all signals in the 'Signal Viewer' and then clicking the 'Add Signal to Viewer' icon  in the 'Simulation Panel'.
- Find signal: 'abb_irb64_24_12_programNumber' within the 'Simulation Panel'. Under the 'Forced Value' column enter '10' (this was the 'Path#' that you entered for the 'abb_irb64_Weld_Op' Robot Program). Ensure that the 'Forced' column is checked.
- Repeat the same process for the 'kr30_3_programNumber' signal, entering the forced value number as '20'. The two signals that you have entered forced values for can be seen highlighted below:



Simulation	Inputs	Outputs	LB	Forced	Forced Value
 abb_irb64_24_12_startProgram				<input type="checkbox"/>	
 abb_irb64_24_12_programNumber		0		<input checked="" type="checkbox"/>	10
 abb_irb64_24_12_emergencyStop				<input type="checkbox"/>	
 abb_irb64_24_12_programPause				<input type="checkbox"/>	
 abb_irb64_24_12_programEnded				<input type="checkbox"/>	
 abb_irb64_24_12_mirrorProgramNumber	0			<input type="checkbox"/>	0
 abb_irb64_24_12_errorProgramNumber				<input type="checkbox"/>	
 abb_irb64_24_12_robotReady				<input type="checkbox"/>	
 abb_irb64_24_12_at_HOME				<input type="checkbox"/>	
 kr30_3_startProgram				<input type="checkbox"/>	
 kr30_3_programNumber		0		<input checked="" type="checkbox"/>	20

- It is now necessary to save the configuration of the 'Simulation Panel'. This can be then reloaded whenever you open the study from scratch. Saving a configuration in this way will save the 'Forced Value' entries for the Robot Programs, else these will need to be re-entered every time the software is closed and opened again. It will also save the order that the signals are displayed in should you want to adjust this. To save a configuration click the 'Store Signals Settings' icon within the 'Simulation Panel' as shown by the red circle.



Enter a name of choice, for example: 'Tutorial_Factory' and click 'OK'.



To open a saved configuration, click the 'Load Signals Settings' icon, see blue circle above, and select the appropriate file.

Note: If you add/ remove signals, it will be necessary to resave the configuration and overwrite the previous save.


It is now required to create additional logic that will automatically initiate these 'Robot Programs' at the correct moment of the simulation. This will be conducted using 'Modules'.

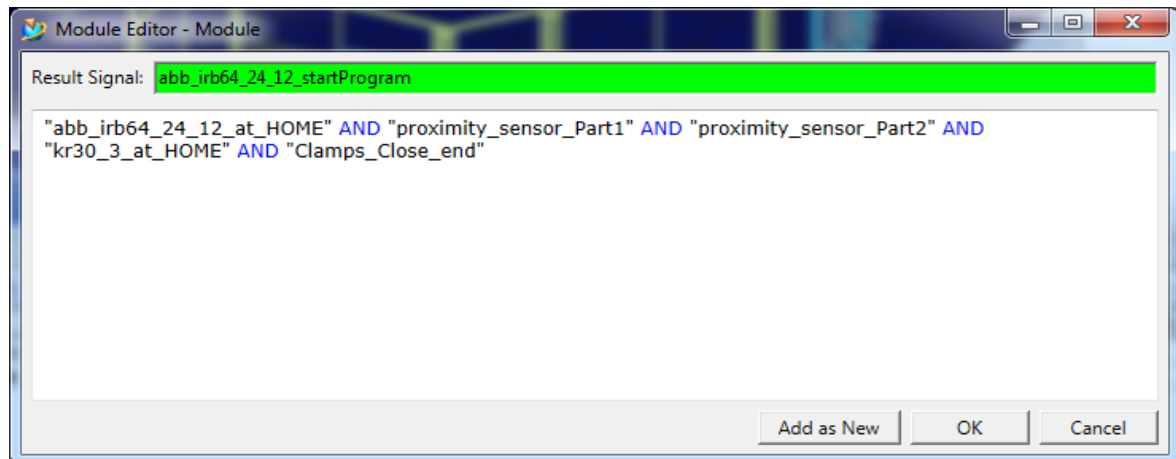
19.3 MODULES

'Modules' contain signals which are a result of logical expressions comprising of a number of other signals and operators. They will be used to govern when the 'abb_irb64_Weld_Op' and 'kr30_3_Pick_And_Place_Op' Robot Programs will start. This process is detailed as follows:

1. Navigate to View >> Viewers >> Modules Viewer.
2. Click the 'New Module Object'  button. Then select 'Module' from the 'Modules Inventory' and click the 'Edit Module'  button.

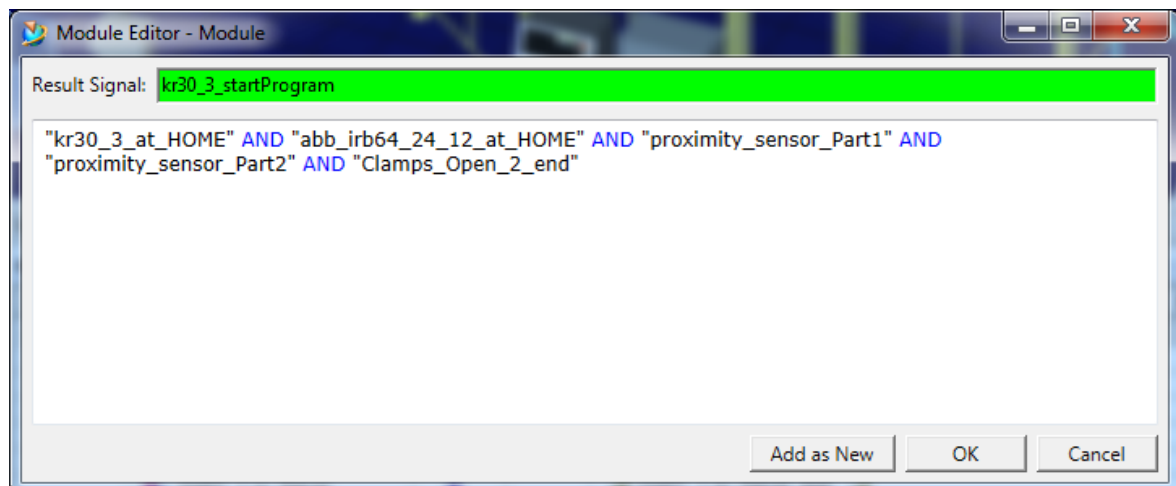
Note: For best practice don't use the 'Main Module'. It is not applicable in this particular tutorial however in other projects it means that you are able to merge studies easily.

3. In the 'Module Editor' click the 'New Entry'  button, this will open a new window.
4. In this new window click on the 'Result Signal' task bar. Open 'Signal Viewer' and double-click 'abb_irb64_24_12_startProgram', this will appear in the 'Result Signal' task bar and turn green. Underneath this task bar, logic should be added to describe the starting condition for the 'abb_irb64_Weld_Op Robot Program'. Input the signals as follows:



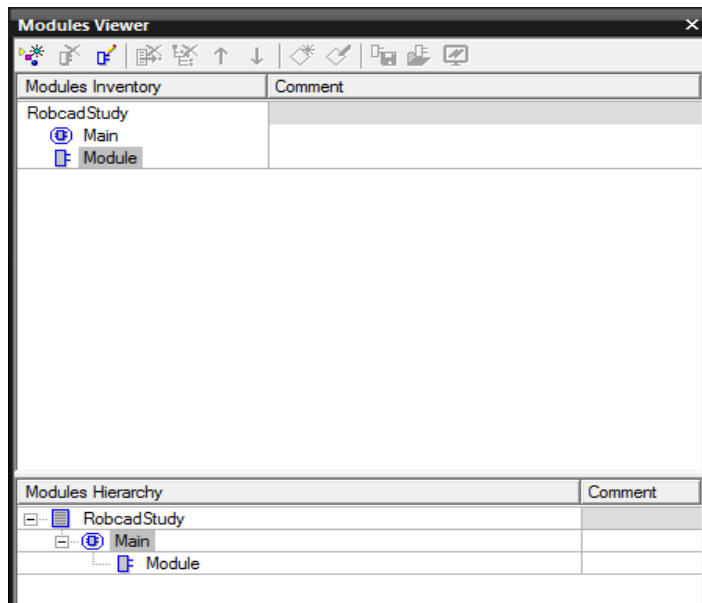
Click 'OK' to close this dialogue box.

5. Again click 'New Entry' within the 'Module Editor' box to add logic for the 'kr30_3 Robot Program'. Enter the 'Result Signal' and logic as follows:



Again, click 'OK' to close this dialogue box. Select 'Close' to return to the 'Modules Viewer'.

6. Now drag 'Module' from the 'Modules Inventory' into the 'Modules Hierarchy'. The 'Modules Viewer' should now look like the following:



At this stage, the simulation can be described as event-based. Two methods of logic have been added: 'Transitions', and 'Modules', each with their own distinct applications.

Transitions: 'Transitions' are used for logic that is specific to an Operation. For the purpose of this tutorial it was decided that this would be the best way to introduce the user to entering logic into the simulation as they are user-friendly and easy to visualise in the 'Graphics Viewer'.

Modules: The 'Modules' are used in this simulation for overall control; they govern the initiation of the 'Robot Programs' and are also used for safety mechanisms (e.g. 'Emergency Stop Button').

Note: The third method of applying logic is through the use of Logic Blocks. These will be added in the next section of this tutorial.

You are now ready to play the simulation. Open the 'Simulation Panel' and press play. You will need to press the 'START CYCLE' signal in the 'Simulation Panel' to start the simulation.

Note: The simulation will run the same as the time-based simulation, however as it is now event-based it will run off the logic.

The next stage in the process is to add safety features to the virtual factory.

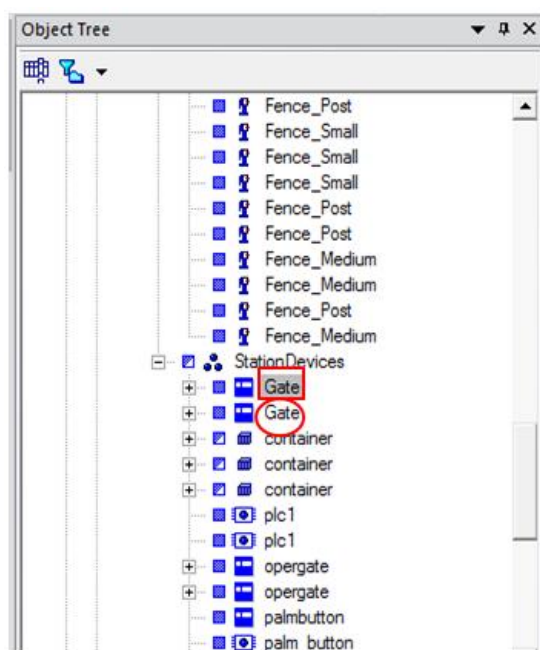
20.0 INTRODUCING SAFETY FEATURES

20.1 GATES AND SAFETY MAT

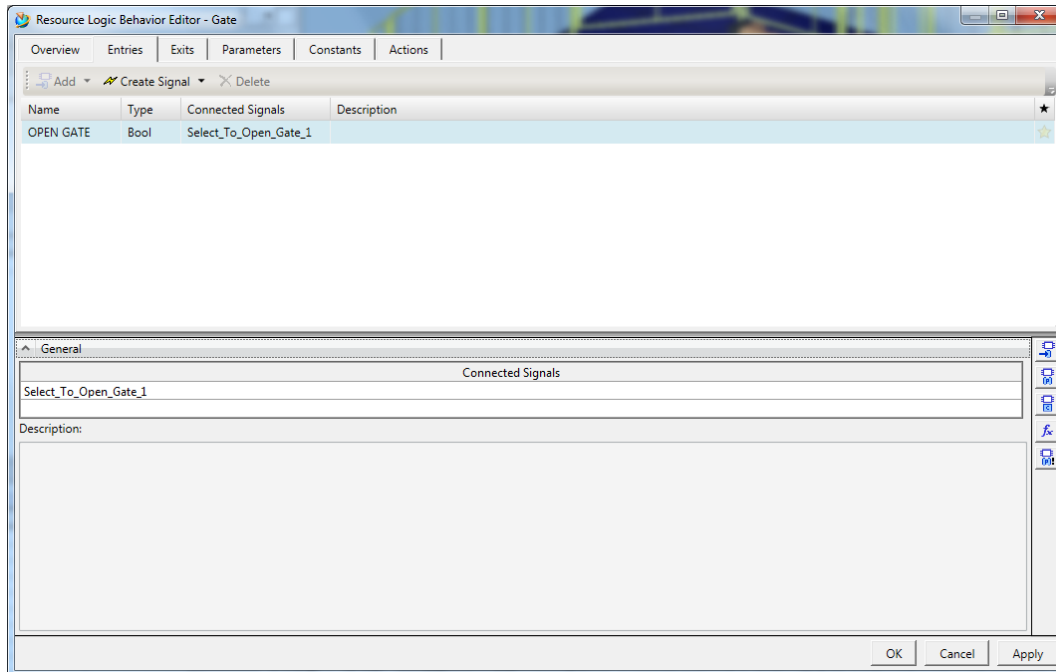
Including 'Gates' to the simulation makes the model more realistic, showing entry points for workers for robot and device maintenance or general access. These 'Gates' must be modelled correctly such that when they are opened the processes are stopped or paused. In addition to this a 'Safety Mat' was introduced to the simulation when an operator was near the clamp Fixture. The 'Gates' and 'Safety Mat' used in this tutorial are 'Smart Components'; this means that logic is applied to these components by means of a logic block (defined logical behaviour derived from one or more specified inputs and outputs in an equation or formula). The main advantages of a 'Smart Component' are that it has predefined actions that signals are connected to and can be stored in a library for use in other projects.

The 'Smart Components' will be defined as follows:

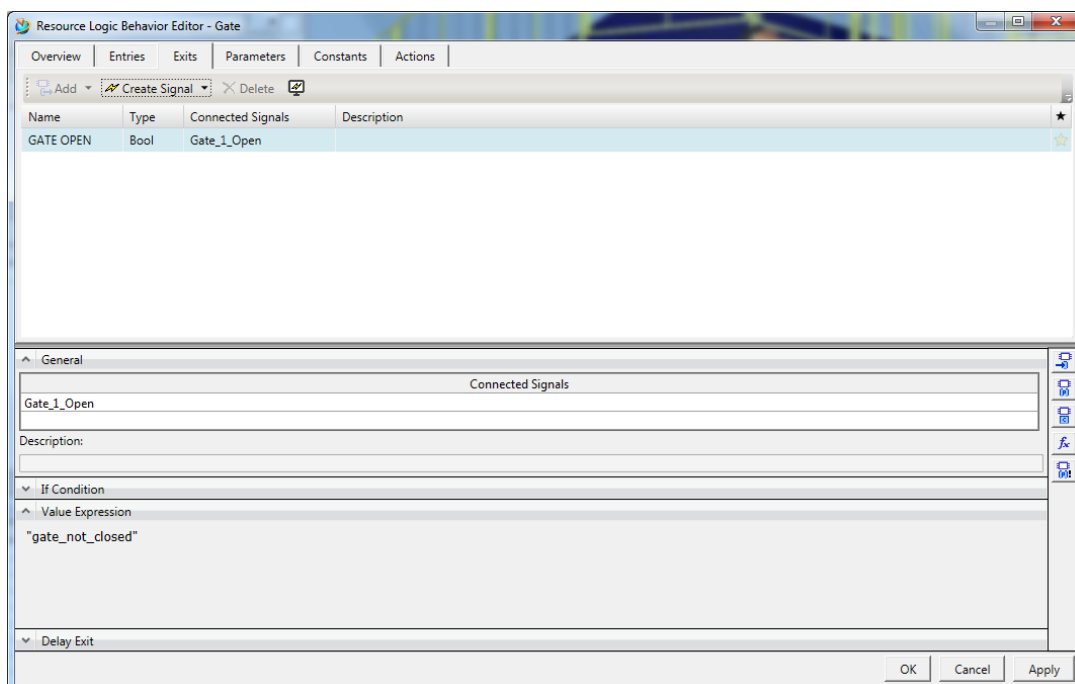
1. Within the 'Signal Viewer', select 'Create New Signal' button. A dialogue box will appear, check the 'Key Signal' box and press OK.
2. Open the 'Signal Viewer' and locate the newly created 'Key Signal', it will be highlighted. Double-click and rename the 'Key Signal': 'Select_to_Open_Gate 1'.
3. Select 'Create New Signal' again. This time check the 'Display Signal' box and press OK.
4. Find the 'Display Signal' within the 'Signal Viewer' and rename the signal: 'Gate_1_Open'.
5. Within the 'Object Tree' under 'StationDevices', select the 'Gate' shown in the red rectangle in the following figure:



6. Whilst the 'Gate' in the red rectangle shown above is still selected, navigate to CEE >> Logic Block >> Edit Logic Resource.
7. Within the 'Entries' Tab, navigate to the 'Connected Signals' box and attach the 'Select_To_Open_Gate_1' signal by selecting it in the 'Signal Viewer' to display the 'Logic Block' window shown below.



8. Within the 'Exit' Tab, again in the 'Connected Signals' box attach the 'Gate_1_Open signal'. Now click 'Apply' and 'OK'. Again this 'Logic Block' window is shown below.

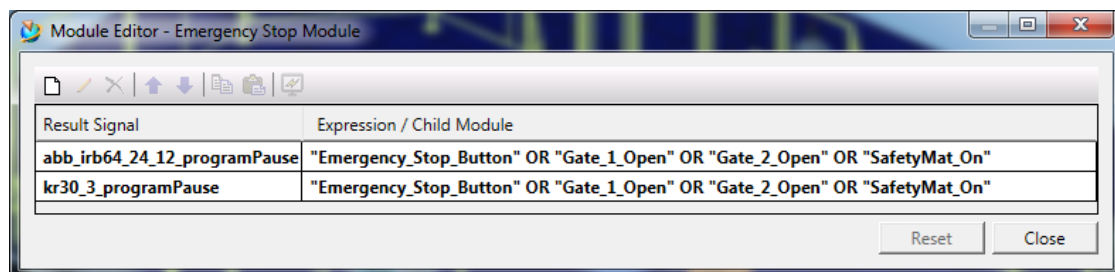


9. Repeat steps 1-8 for the second 'Gate' shown in the above figure in the red circle, Naming the 'Key' and 'Display Signals' as 'Select_to_Open_Gate_2' and 'Gate_2_Open' respectively.
10. Repeat steps 1-8 for the 'Safety Mat' named 'safetymate' in the 'Object Tree'. Naming the 'Key' and 'Display Signals' as 'Human_on_SafetyMat' and 'SafetyMat_On' respectively.

20.2 EMERGENCY STOP BUTTON

In order to mimic a real life factory, safety measures and an 'Emergency Stop Button' should be included. The conventional 'Emergency Stop Button' can be used to stop the simulation at any point and return the Robots to their home positions, however, for the purpose of this basic tutorial the 'Emergency Stop Button' will simply be used to pause the simulation.

1. Within the 'Signal Viewer', create another new 'Key Signal' and rename this: 'Emergency_Stop_Button'.
2. Open 'Modules Viewer' and create a new 'Module'. Double-click the newly created 'Module' and rename: 'Emergency Stop Module'.
3. Edit the 'Module' by creating two new entries containing the following parameters:

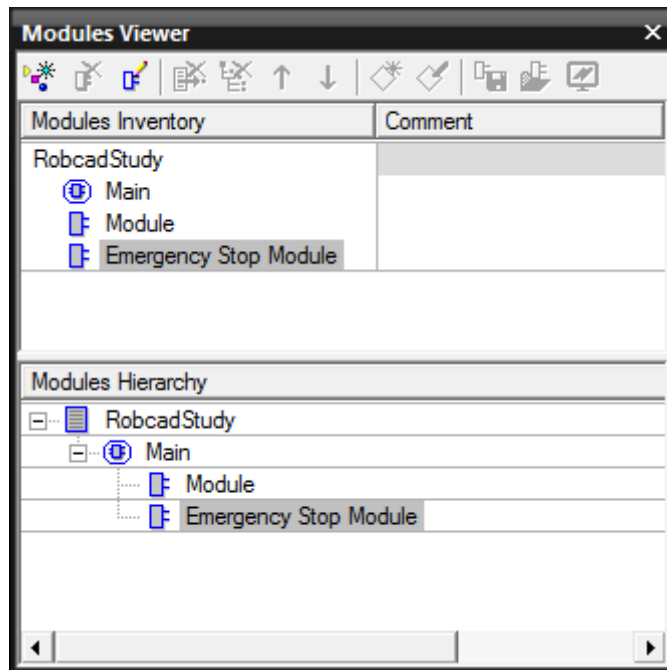


Note: The 'Emergency Stop Module' will pause the simulation on three possible situations:

- The 'Emergency Stop Button' is pressed: This is representative of the stop button in a real factory being pressed by an operator.
- One or both of the 'Gates' are open: This safety feature is representative of an operator entering the work zone during the process.
- The 'Safety Mat' sensor is triggered: This is representative of a human being present on the mat during the process.

All three of these situations would need to be included in a real factory to ensure the safety of the workers/ operators.

- Click 'Close' and drag the 'Emergency Stop Module' into the 'Modules Hierarchy', as shown below:



- Once again select all the signals in the 'Signal Viewer' and add them to the 'Simulation Panel'.
- The simulation is now complete and you are free to run it using the play button in the 'Sequence Viewer'.

Note: To implement any of the safety features that have been added to the simulation the user must manually trigger the signals. This is performed using the 'Simulation Panel'. By checking the 'Forced' column for either the 'Emergency_Stop_Button', 'Select_To_Open_Gate_1', 'Select_To_Open_Gate_2' or 'Human_on_SafetyMat' signals the simulation will pause. Unchecking the 'Forced' column will resume the simulation.

21.0 PRACTICE ACTIVITY

The tutorial is now complete and you should now have a fully operational virtual factory that runs as a discrete event-based simulation. As an extension task, the 'opergate', 'panelview' and 'RobotController' devices are pre-programmed as 'Smart Components'. Different poses are set for the lights on the respective devices, i.e. for one pose the light is red and for an alternative pose the light is green. This can be previewed by right-clicking the component and selecting 'Pose Editor'. You should introduce the 'Smart Component' for each device into the



simulation by creating a new 'Module' and setting the conditions so that the lights change colour based on the stage the simulation is at.

B

Oculus Rift - User and Development Guide

1 Directory Structure

Our project package Oculus_Dir.zip contains the following subdirectories:

ActiveTcl	ActiveTcl installer version 8.6.4 for windows (x86).
AutoHotKey	AutoHotKey installer that is used to compile our application.
Jack	Jack 8.2 Windows (x64) installer and licence file.
Oculus Runtime	Oculus Runtime for Windows, version 0.4.4-beta.
OculusJack	Source code for our custom application and compiled version.
Oculus	Source code for our custom Tcl module for Jack 8.2.
OculusSDK	Oculus SDK package version 0.4.4 that is used to develop applications.
OculusSDK\LibOVR	Libraries, source code, projects, and makefiles for the SDK.
OculusVR Source	Source code for OculusVr-CmDLL Project
TCL	Root folder from installation of ActiveTcl 8.6.4
tclovr	Source code for TCL-DLL Wrapper with all additional required files.
Virtual Desktop	Virtual Desktop Windows (x64) installer for Oculus Runtime 0.4.4.

2 Set up Software

2.1 Connection between Oculus Rift and Computer

1. Install Oculus Runtime (version 0.4.4) from the "Oculus Runtime" folder.
2. Copy the "OculusSDK" folder into the root directory of the installed Oculus Runtime.
3. Read the DK2 Quick Start Guide and connect Oculus Rift to the computer.
4. Restart the computer and run the Oculus World Demo in the "OculusSDK" folder in order to see if Oculus Rift is working.

2.2 Install Jack

1. Install Jack (version 8.2), jack82-win64-inst.exe including all plugin addons from the "Jack" folder.
2. Copy the licence file from the "Jack" folder into the licence folder of the installed root directory of Jack 8.2.

2.3 Install our Oculus module in Jack

1. Open the root directory of Jack 8.2 and go to library -> modules. Full path on Windows 7 might be C:\ProgramFiles\Siemens\Jack_8.2\library\modules.
2. Copy the "Oculus" folder into the modules folder.

2.4 Install Virtual Desktop

1. Install VirtualDesktop-Setup.x64 from the "Virtual Desktop" folder.
2. Restart computer.

3 Set up Environment

3.1 Import Human Object into Process Simulate

1. Create a new project via the AdminConsole.
2. Open it with Process Design and create Libraries, Studies and Human.
3. Load HUMAN_MODELS from C:\Program\Files\Tecnomatix\emPower\Human\HUMAN_MODELS into Libraries.
4. Create a RobCadStudy in Studies.
5. In Human, create a ProcessResource with a jack (named jack) from HUMAN_MODEL in Libraries.
6. Open Process Simulate via PS Standard Mode in RobCadStudy.
7. Now you should see the Jack in the scene.

3.2 Connection between Human Object in Process Simulate and Jack 8.2

1. Follow the "Import Human Object into Process Simulate" step.
2. Create a human object named "jack" (it must has the same name as the human object in Process Simulate). This can be done by loading the Oculus module from the plug-ins menu and choose "Generate jack".
3. Load JackCollaboration from the plug-ins menu.
4. Choose server and click on start.
5. Open up JackCollaboration in Process Simulate and choose client and click connect.
6. Choose jack as figure in add figure in Jack.
7. Now jack should be displayed as connected in Process Simulate.

4 Set up Orientation Control and Rendering to Oculus Rift

4.1 Set up the Orientation Control

1. Follow the “Connection between Human Object in Process Simulate and Jack 8.2” step.
2. Load Oculus module from plug-ins menu.
3. Choose Movement...
4. Start and run the custom application JackOculus.exe from the "JackOculus" folder.
5. Connect a GameCube controller to the computer if you want to use one instead of the keyboard.

4.2 Render Graphics from a Process Simulate Scene to Oculus Rift

1. Follow the “Set up the Orientation Control” step.
2. Make sure that VirtualDesktop x64 is installed.
3. Set rift display mode to “extend desktop to the HMD” in Oculus runtime (you may get one or more BSODs if you are unlucky).
4. If the image is rotated in the Oculus Rift goggles, rotate it to its correct orientation by right clicking the desktop, select screen resolution and set the rift DK2 orientation to portrait.
5. Open Process Simulate and Jack 8.2 and configure the left and right eye view windows in Process Simulate. This is done by right clicking the human object "jack" in Process Simulate, choose vision window and select left and right eye.
6. Open Virtual Desktop and choose the F8 mode (To rift - side by side content)
7. Run JackOculus.exe if not already started. Press F12 key to toggle between screen modes.
8. Wear Oculus Rift, you should now see the Process Simulate scene from Jack’s eyes. You should also be able to control the movement of the human object with either your GameCube controller or by keyboard arrow keys.

5 Development

5.1 Compile our Custom Application

1. Install AutoHotkey112200_Install.exe from the "AutoHotKey" folder.

2. Press start icon -> All Programs -> AutoHotKey -> Convert .ahk to .exe
3. Browse source (script file) and choose folder "OculusJack", file Oculus-Jack.ahk
4. Browse destination (.exe file) and choose the name OculusJack.exe.
5. Press the >Convert< button

5.2 GameCube Device Key Names

After running the JoystickTest.ahk script described in the methodology, following table was established with all GameCube controller device key names determined:

GameCube button	Device key name
X Button	Joy1
A Button	Joy2
B Button	Joy3
Y Button	Joy4
L-Trigger	Joy5
R-Trigger	Joy6
Z Button	Joy8
Start/Pause Button	Joy10
D-pad up	Joy13
D-pad right	Joy14
D-pad down	Joy15
D-pad left	Joy16
GameCube button	Key state
L-Trigger analog	JoyV
R-Trigger analog	JoyU
Control Stick	JoyX, JoyY
C-Stick	JoyZ, JoyR

After experimenting with the JoystickTest.ahk script it was concluded that the key states JoyU, V, X, Y, Z and R can take values between 0 and 90 on a GameCube controller.

5.3 Compile the Tcl-Dll Wrapper in Visual Studio

1. Open Visual Studio
2. Choose File -> New Project
3. Choose Win32 Console Application, set name and solution name to tclovr and choose the "tclovr" folder as location.
4. Press next, choose application type: DLL and additional options: Empty project
5. Press finish

6. Drag OculusVr-CmDLL.cpp, OvrThrd.cpp, UdpClient.cpp, OvrClient.cpp and stdafx.cpp files into Source File folder in the project
7. Drag the OculusVr-CmDLL.h file into Header Files in the project
8. Right click on tclovr project -> Properties -> Configuration Properties -> C/C++ -> General
9. In Additional Include Directories, include the following folders found in the "tclovr" folder: TclWin, LibOVR\Src, LibOVR\Include and Tcl
10. Head to Configuration Properties -> C/C++ -> Preprocessor
11. In Preprocessor Definitions, use the following symbols: WIN32, NDEBUG, _WINDOWS, _USRDLL, OCULUSVRCMDLL_EXPORTS, USE_TCL_STUBS and _WINSOCK_DEPRECATED_NO_WARNINGS
12. Head to Configuration Properties -> Linker -> General
13. In Additional Library Directories, include following folders found in the "tclovr" folder: LibTcl and LibOVR\Lib\Win32\VS20XX where XX is the Visual Studio version you are compiling with
14. Head to Configuration Properties -> Linker -> Input
15. In Additional Dependencies, add: libovr.lib, libovrd.lib, winmm.lib, tclstub86.lib and ws2_32.lib
16. Press apply and close properties menu
17. Build -> Build solution
18. You should now find the compiled file tclovr.dll in the Release folder.

C

Inventory of Safety Equipment

Inventory List of Safety Equipment

Product	Description	Quantity	Relevant
Focus light grid FTR4-K1C-500	Optical transmitter and receiver units. Beams of infrared light are sent to the receiver from the transmitters. When a light beam is interrupted, a dual stop signal is given to the dangerous machines inside the Light Curtain/Grid protected area	1	Yes
Focus light curtain FT4-35-300	See Focus light grid FTR4-K1C-500	1	Yes
Smile Tina 11 EA	Emergency stop button with LED indication designed to be installed in areas with space limitations	7	Yes
JSHD4 Three-Position Enabling Devices	Device with double three-position button, gives a stop signal when released or fully pressed in, used for troubleshooting, programming and test running	2	No
JSTD1 Safeball	Control device that is used to guarantee that the operator's hands will be kept outside a risk area.	1	No
Eden sensor	Non-contact switch with a dynamic function	7 pairs	Yes
Grey emergency stop	Emergency stop with reset and access buttons, probably not ABB JOKAB equipment	2	No
Pluto B46-6	Safety PLC	1	Might be
Pluto B20	Safety PLC	1	Might be
Pluto BT51	Safety relay, used to supervise safety devices	3	Might be
Electromagnetic door lock	Prevents doors and gates from opening	2	No
Pluto Gateway Profibus	Gateway for communication with other PLC's through Profibus	1	Might be
Telemecanique relay	Relay from Telemecanique	4	No
Spot safety light beam	Photoelectric guarding of an entrance or around a risk area	4 pairs	Yes
M12-3A Y-branch	Branch contact for reducing risk of incorrect connections	4	Yes
M12-3B Y-branch	See M12-3A Y-branch	2	Yes
TINA 10A contact	Connection unit with M12 connections, that make it easy to connect a light curtain or light beam Focus with OSSD outputs to the dynamic safety circuits of Vital and Pluto	2	Yes
TINA 1A contact	Device used as a blind plug in unused M12 connections	4	Yes
TINA 2A contact	Device that adapts the safety sensors with mechanical contacts, such as emergency stops, switches and light curtains/light grids, with their own relay outputs to the dynamic safety circuit	2	Yes
TINA 4A contact	Tina 4A is a connection block with four 5-pin M12 connections	1	Yes

D

Tutorial: Setting Up a Project in TIA Portal

Setting Up a Project in TIA Portal

May 2015

Introduction

This is a guide that clarifies the process of establishing a PLC setup in TIA portal. The equipment used are listed below:

Software

- SIMATIC STEP 7 Professional V13 Service Pack 1
- SIMATIC NET V13

Hardware

- SIMATIC 1517F-3 PN/DP, CPU
- PM 70W 120/230VA, Power supply
- IM 155-6 PN ST, I/O-device
- DQ 8x24VDC/0.5A ST, Digital output module
- DI 8x24VDC ST, Digital input module
- General server module
- Netgear Gigabit Switch ProSAFE—GS105

Guide

This is a step by step guide. It is recommended to save in between every step to avoid losing progress and to confirm the steps are done in the correct order, they are all essential to achieve a functioning configuration.

Creating a project

Step 1

In the portal view of TIA portal, choose "Create new project" [1] as seen in Figure 1. Name the project and click on create [2]. TIA portal will then establish a clean project worksheet.

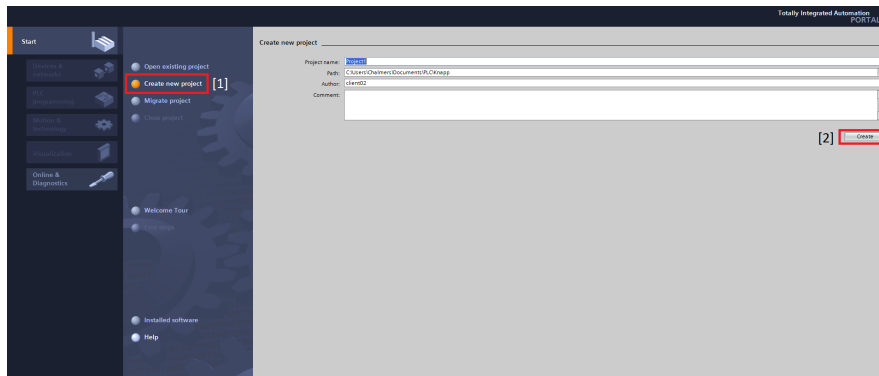


Figure 1: Creating a project

Step 2

When the loading is finished, press "Project view" [3] in the lower left corner of the screen, as seen in Figure 2, this will bring up the project view.

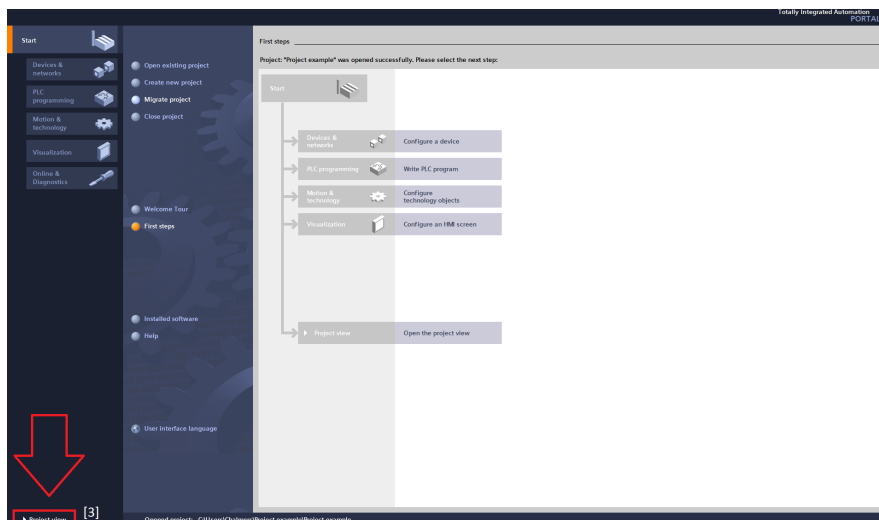


Figure 2: Project view

CPU configuration

Step 3

Press the "Add new device" [4] button at the top of the project tree and choose the controller section [5]. In this project we will be using a Siemens PLC of the 1500 series, select the "Simatic S7-1500" folder followed by "CPU 1517F-3 PN/DP" and the "6ES7 517-3FP00-0AB0" [6], as seen in Figure 3. Make sure the "Open in device view" box at the lower left corner of the window is checked. Make sure the firmware version displayed in the "Version" drop-down menu matches the firmware in the PLC. This can be seen on the display of the PLC.

These numbers represents the PLC serial number and can be found under the display cover [7], as seen in Figure 4. As the display itself and each attached device to the CPU has its own serial number, be sure to check the number corresponding to the CPU.

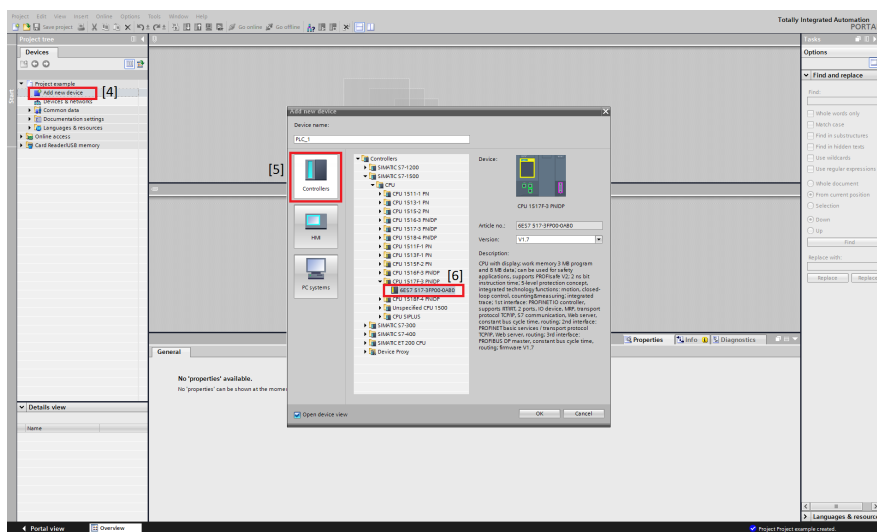


Figure 3: Selecting PLC

The PLC CPU and its rail can now be seen in the device view, where the CPU can be seen below "PLC_1" in slot 1. The slots represent devices connected to the CPU according to the physical setup. As seen in Figure 4, the CPU is connected to a power supply module, seen to the left of the opened lid [8]. Next the power supply will be added into the corresponding configuration slot. The serial number of the power supply module can be found at the top of the device [9], as seen in Figure 4.

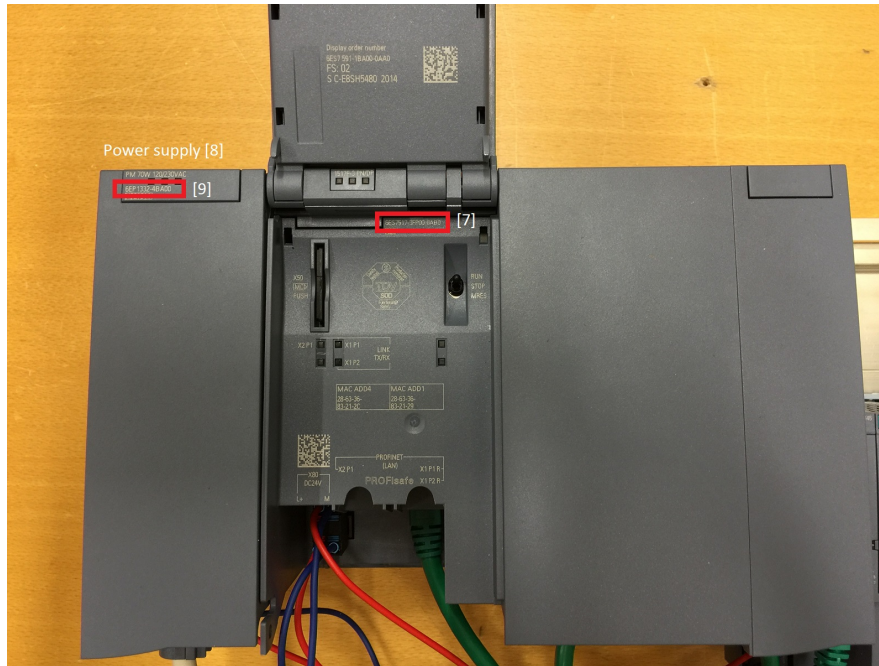


Figure 4: PLC serial numbers

Step 4

In the catalog menu of TIA portal [10], seen to the right in Figure 5, select the "PM" folder at the top. This is the folder for the power supply modules. In this project the "PM 70W 120/230VA" is used, and the corresponding serial number "6EP1332-4BA00" is selected. Drag and drop the selected power supply module in to "slot 0". The setup should now look as in Figure 5.

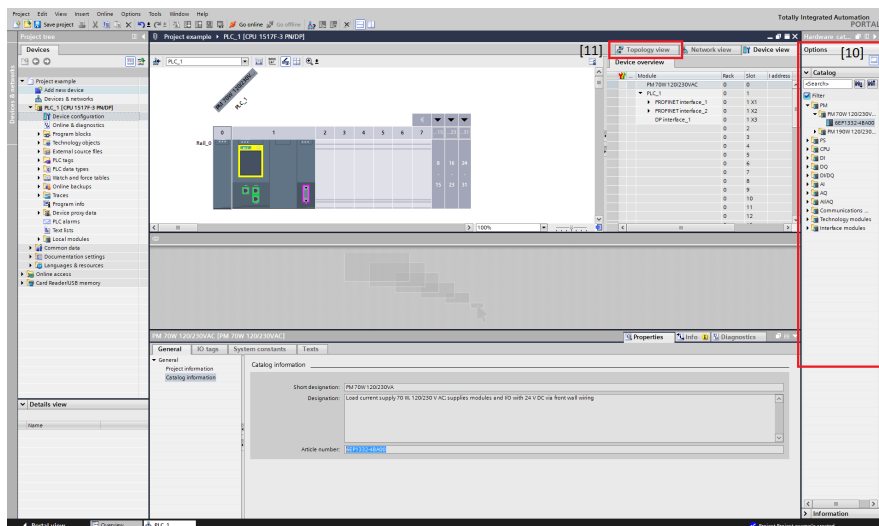


Figure 5: PLC CPU setup

I/O device configuration

Step 5

Next the I/O device will be added to the configuration. In order to do this, toggle to the topology view, by selecting the corresponding tab on top of the device view window [11], seen in Figure 5.

The tabs displayed are the viewing options. The device view is used when configuring a specific module, the topology view is used for adding hardware to the configuration and the network view will be used for setting up the communication between the devices.

When in the topology view, notice that the catalog menu has been updated to add controllers, HMI interfaces, PC systems etc. These represent the main components of the configuration, into which modules can be integrated.

Step 6

It is possible to find a specific component by entering its serial number in the search bar at the top of the folder view in the catalog menu [12], seen in Figure 6. The serial number of the I/O interface can be found on top of the interface [13], as seen in Figure 7.

Type the serial number "6ES7 155-6AU00-0BN0" into the search bar or click open "Distributed I/O", "ET 200SP", "Interface modules", "PROFINET" and "IM 155-6 PN ST" [14]. Finally, drag and drop the "6ES7 155-6AU00-0BN0" I/O device into the topology view, the configuration should now be the same as in Figure 6.

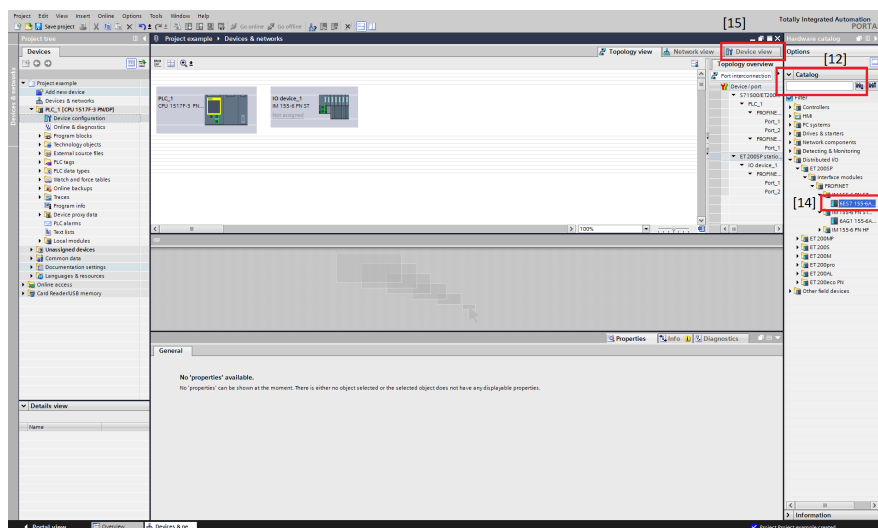


Figure 6: Adding I/O device

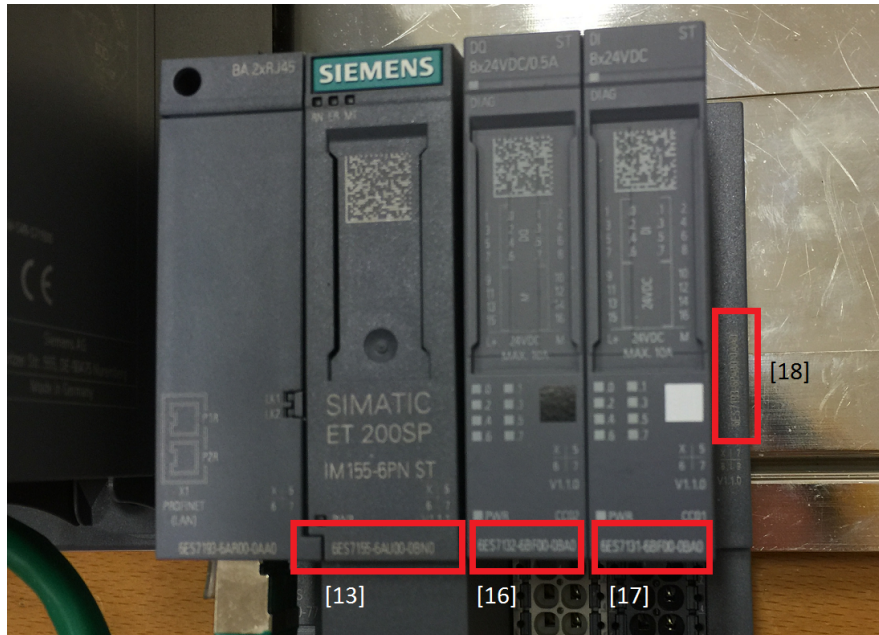


Figure 7: I/O device serial numbers

Step 7

All main components are now represented in the main configuration. Next, enter the device view of the I/O interface by selecting it in the topology view followed by switching to device view in the tab as described before [15], see Figure 6.

Step 8

Enter the serial number of the digital output module "6ES7 132-6BF00-0BA0" [16], the digital input module "6ES7 131-6BF00-0BA0" [17] and the server module "6ES7 193-6PA00-0AA0" [18] in the catalog menu search bar respectively. The modules can also be found by browsing the catalogue [19], see Figure 9. Drag and drop the devices into the device view, in their corresponding order onto the rail of the I/O device. The configuration should now look as in Figure 9.

If prompted to start the search from the beginning of the catalog, when searching for the components serial numbers in the search bar, click "OK", see Figure 8.

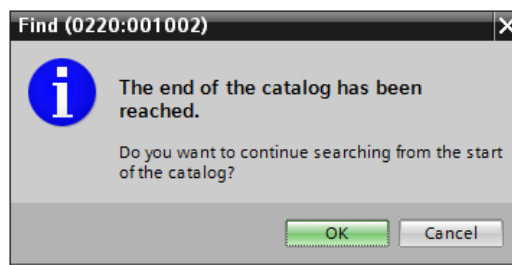


Figure 8: Catalog search

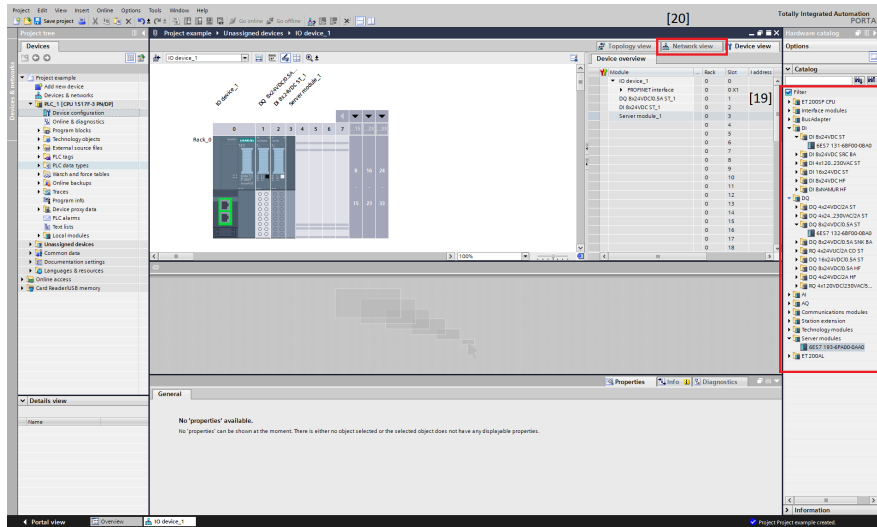


Figure 9: I/O device configuration

Network configuration

The hardware setup is now complete. The next step is to connect the devices through a network and establish a communication link.

Step 9

First, enter the network view [20], by pressing the network tab at the top of the topology window, see Figure 9. When in network view, see Figure 10, notice how the toolbar in the top section of the view has changed. It is possible to switch between "Network mode" and "Connection mode" [21] displayed at the top left corner of the view. In network mode, the devices are connected and in the connection mode, the protocol of these connections can be edited.

The available connection ports are displayed by green and pink boxes. The green boxes are PROFINET interfaces, the one's that will be used in this guide. The pink is a PROFIBUS connection [22]. This connection can be used when pairing multiple CPU:s, but as this setup only includes one, it will not be explored further.

Step 10

The CPU, seen in Figure 10, has two PROFINET interfaces, one of which will be assigned to the I/O device. Select network mode. Press and hold the cursor over the rightmost PROFINET interface [23] and drag the cursor to the PROFINET interface of the I/O device [24]. The I/O device has now been assigned to the PLC CPU.

The line is at this point dashed because of the selection box [25], displayed in the top right corner of the view. This has no major effect and can be undone by deleting the box.

This connection can be seen in Figure 11, as the green Ethernet cable running from the CPU to the IO device [26].

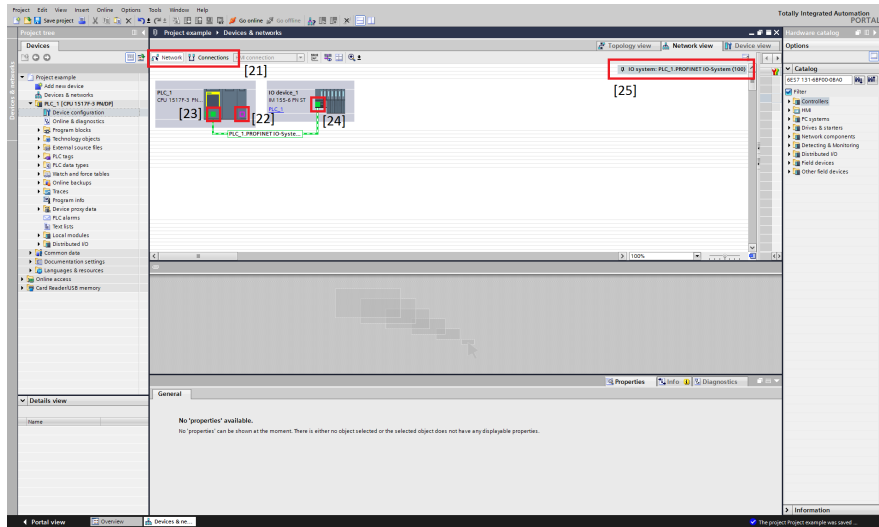


Figure 10: Virtual I/O-device connection

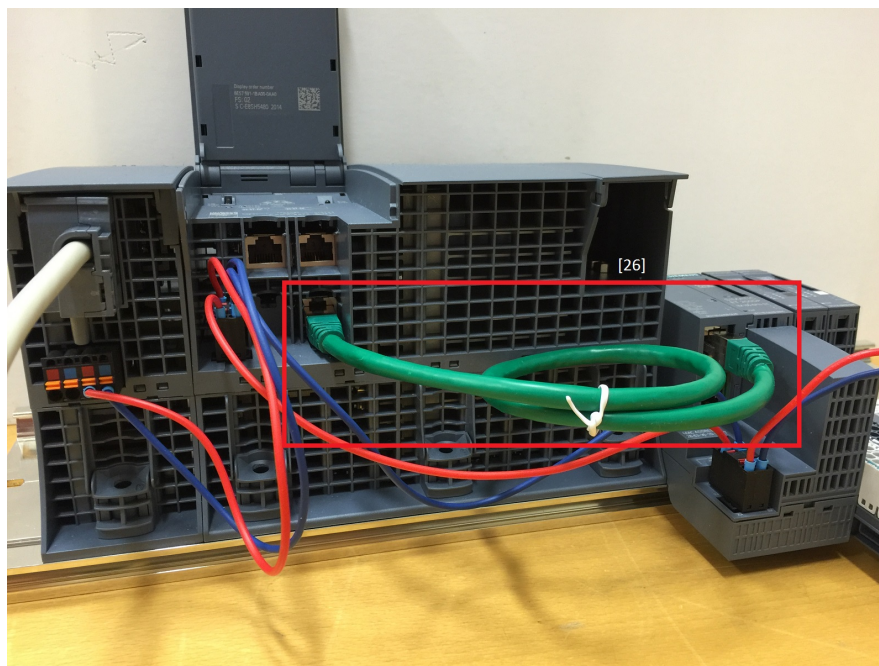


Figure 11: Physical I/O-device connection

Step 11

To connect the PC-station to the CPU, an Ethernet cable is needed. The physical setup is similar to that of the connection between the CPU and the I/O device. In Figure 12, the setup can be seen. Notice that the connection is made to the second PROFINET interface [27], see Figure 13. A switch is used as a connection hub between the devices, this has no major effect and the connection can be made directly.

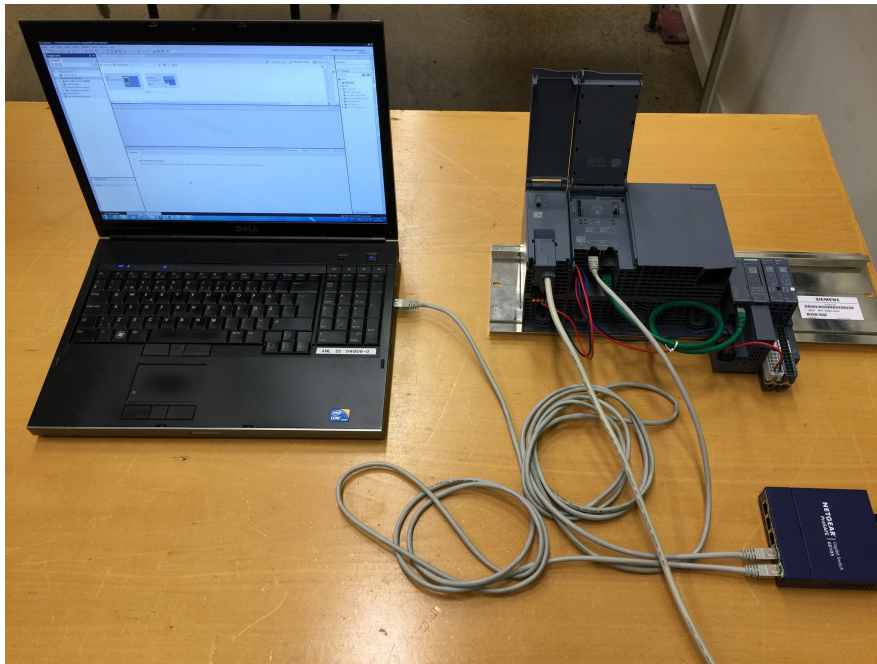


Figure 12: Physical setup

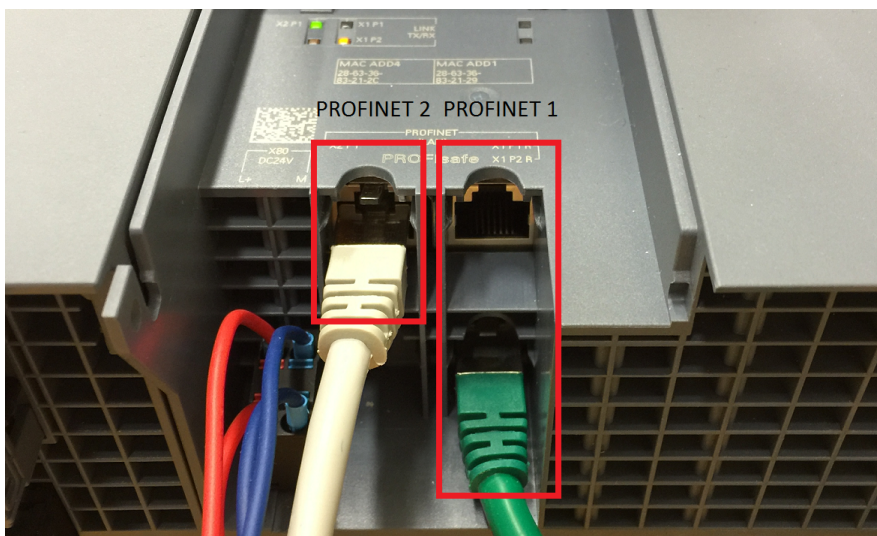


Figure 13: PROFINET interface connections

Step 12

Before the PLC is compiled, turn it on. The power switch is located under the lid of the power supply module, see Figure 14. Once the CPU turns on, wait until the LED lights on the top of the device has stopped blinking and the screen says run [28].



Figure 14: Powering on the PLC

Step 13

In the project tree, right click on the CPU, named as PLC_1. Select "Download to device" followed by "Hardware and software (only changes)" [29], see Figure 15.

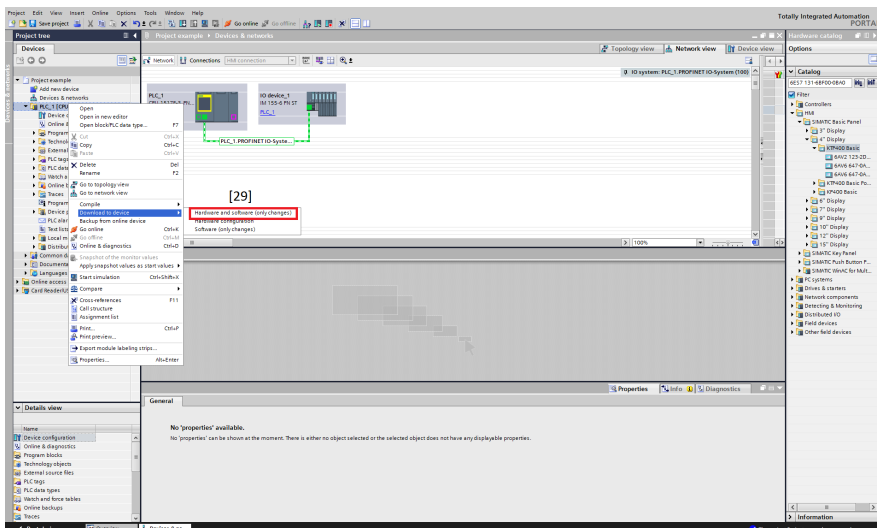


Figure 15: Download to device

Download to device

Step 14

In the extended download to device window that follows, select the PG/PC interface as "PN/IE", next select the "Broadcom NetXtreme 57xx Gigabit Controller" and finally "Try all interfaces" [30], see Figure 16. Press "Start search" [31].

What these options do is to check the communication through the network card on the computer, through the Ethernet interface and to try all the interfaces of the CPU. The connection data can be seen above. Alternatively the IP-address of the CPU can be entered in the "Compatible devices in target subnet" and "Access address" seen in the first line [32], see Figure 16. The CPU software will then only search this specific address for the CPU.

Step 15

If the setup has been done correctly the CPU should become visible as a target. To be sure that the connection is successful press the Flash LED button [33]. At the top of the CPU, above the display, the LED lights should now be blinking. Next press load [34].

If a prompt is shown that requires an extra IP-address, this means the current one is not in the same subnet as the CPU. Adding a new IP-address as TIA portal suggests, might result in difficulties later in the project, for instance in the addition of an OPC server.

In order to complete the next section the IP-address in Windows might have to be altered. To change the IP-address in windows, enter the Control Panel, "Network and Sharing Center" and "Change adapter settings". Right click on "Local Area Connection", this represents the Broadcom NetXtreme 57xx Gigabit Controller, and choose "Properties". Next turn off the Internet Protocol Version 6, select Internet Protocol Version 4 and choose properties. Now change the IP-address so that the subnets match, that is all the numbers match except for the last. In this guide the IP-address of the PROFINET interface 2 is set to the standard "192.168.1.1" and Windows IP-address to "192.168.1.5". See Figure ???. In order for TIA portal to properly register the change of IP-address a computer reboot may be needed.

Next press load [35] and finish [36].

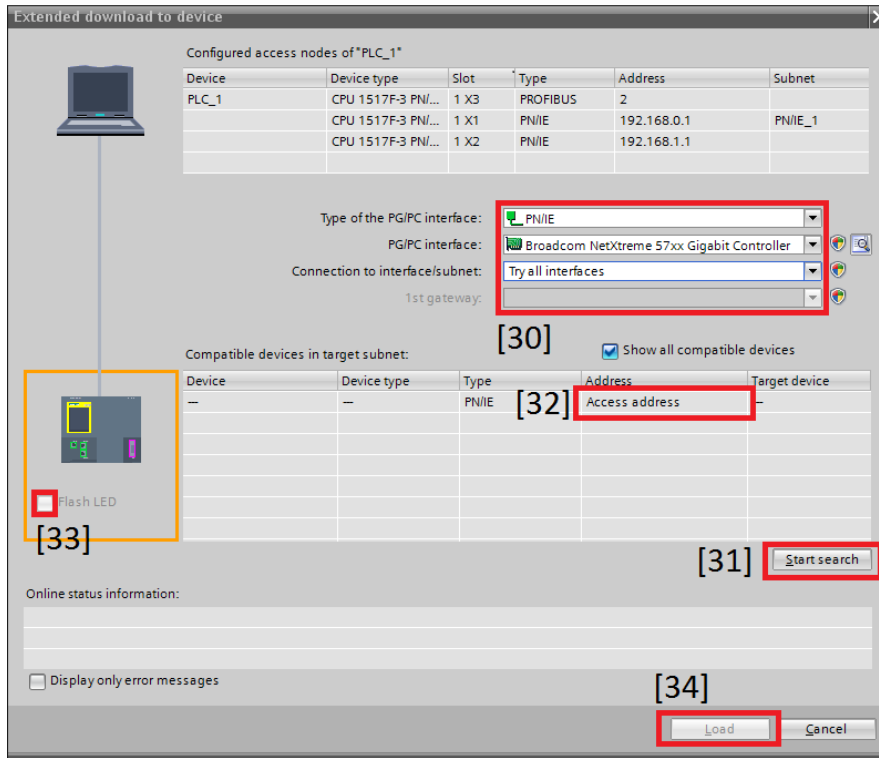


Figure 16: Extended download to device

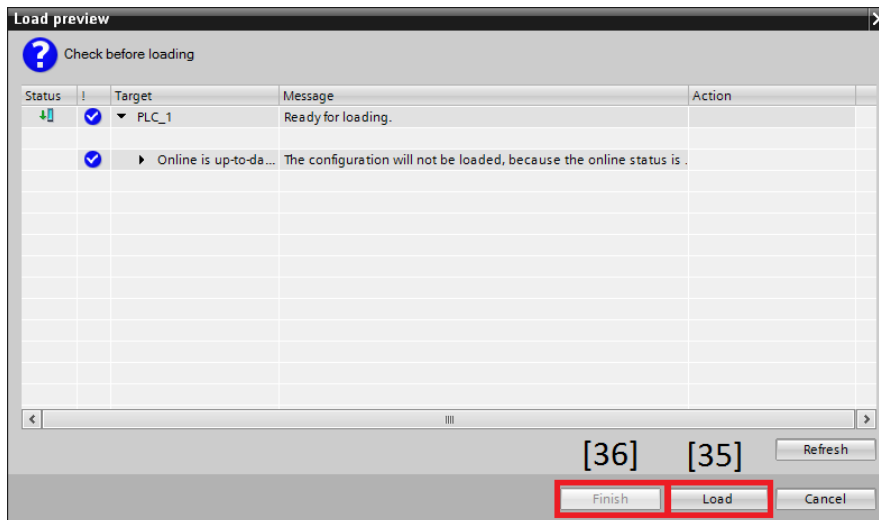


Figure 17: Loading to device

The PLC is now configured to the PC-station. Additional equipment can be connected to the I/O device and programmed by ladder code in TIA portal.

E

Tutorial: Setting Up an OPC Server

Setting Up an OPC Server

May 2015

Introduction

An OPC server is going to be added to the already created PLC configuration, described in the guide "Setting up a project in TIA portal". The server will be running through TIA portal and on Station Configurator Editor. The Station Configurator Editor will pose as the virtual representation of the OPC server and the IE-network card.

Guide

TIA portal setup

Step 1

Select "Add a new device" in the "Project tree" [1], see Figure 1. Select the "PC systems" tag [2] and choose the "PC general" [3], click "OK" [4].

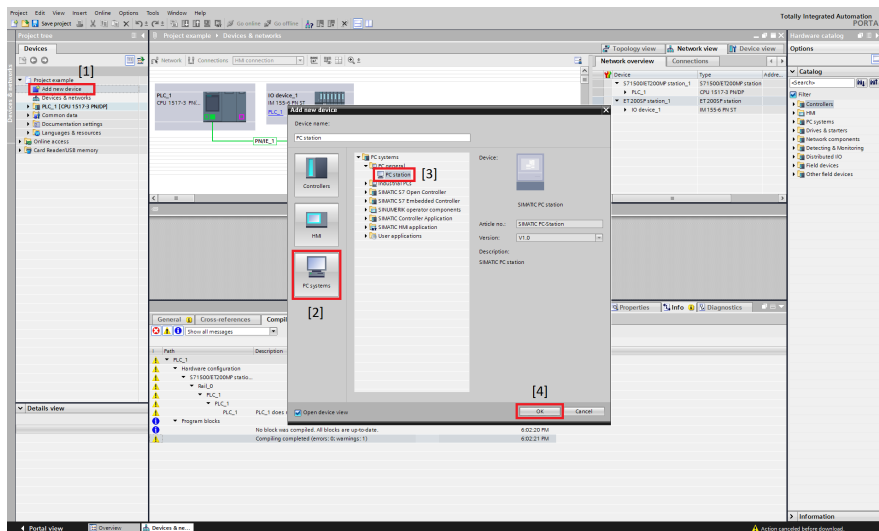


Figure 1: Add PC station

Step 2

In device view of the PC station, add a IE general network card by either searching "IE general" in the search bar of the catalogue menu or by browsing "Communication modules", "PROFINET/Ethernet" and finally "IE general" [5]. Next add an OPC server, either by searching "OPC server" or by browsing "User applications" and "OPC sever" [6]. Drag and drop IE general, followed by the OPC server according to Figure 2.

Right click the IE general network card and select the "PROFINET interface (X1)" section, see Figure 3. Next, enter "Ethernet addresses" and choose the same IP address as Windows. Next turn off the Internet Protocol Version 6, this step is described below.

To change or view the IP-address in Windows, enter the Control Panel, "Network and Sharing Center" and "Change adapter settings". Right click on "Local Area Connection", this represents the Broadcom NetXtreme 57xx Gigabit Controller, and choose "Properties". Next turn off the Internet Protocol Version 6 by unchecking the box in the properties menu. Select Internet Protocol Version 4 and choose

properties. Now change the IP-address so that it corresponds to the IE-network card in TIA portal. The IP-address "192.168.1.5" will be used in this guide. Note that the subnet mask also needs to be the same for all devices.

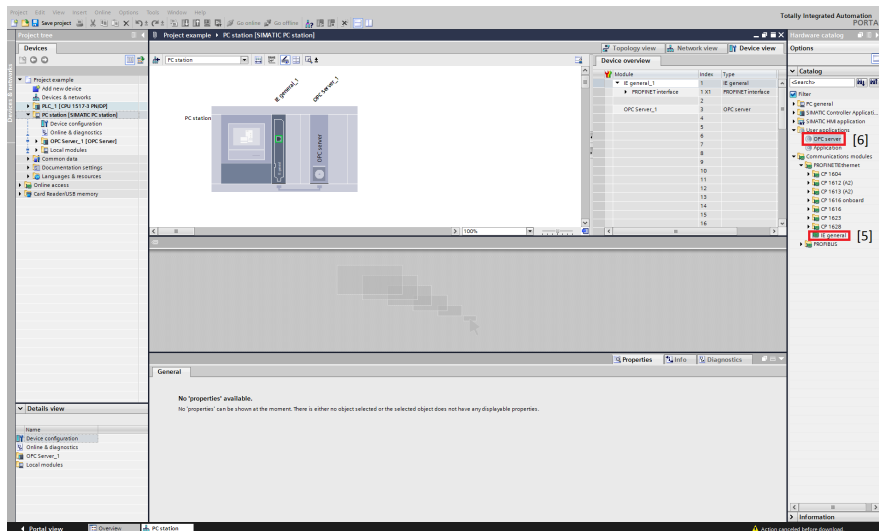


Figure 2: PC station configuration

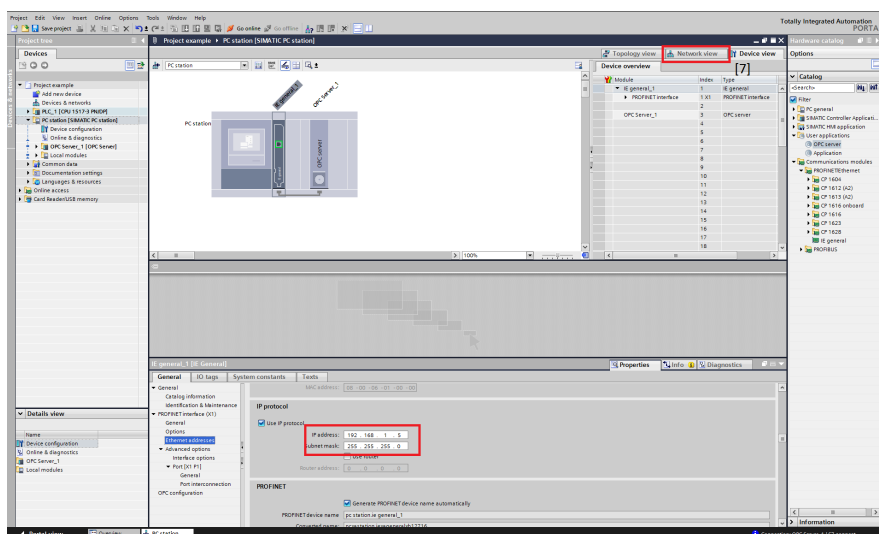


Figure 3: IP-address configuration

Step 3

Enter the network view [7], see Figure 3 followed by "Connection mode" [8], this is done by pressing the button in the top left corner of the network view. Press the down arrow in the drop down menu [9] and choose the "S7-protocol" [10], as seen in Figure 4.

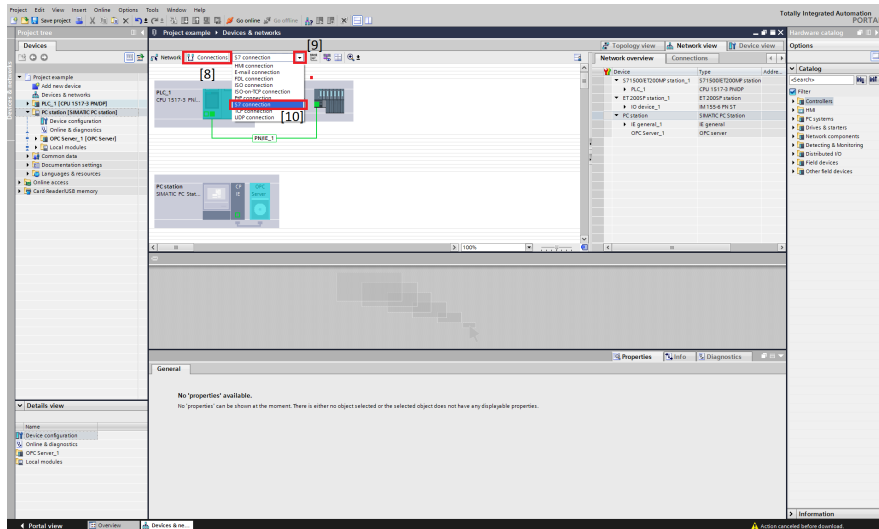


Figure 4: S7-protocol connection

Step 4

Connect the CPU and the PC station by clicking the PROFINET interface on the IE general network card, followed by dragging a connection line to the PLC:s PROFINET interface 2. Move the PC station to the same level as the PLC. The setup should now appear as in Figure 5.

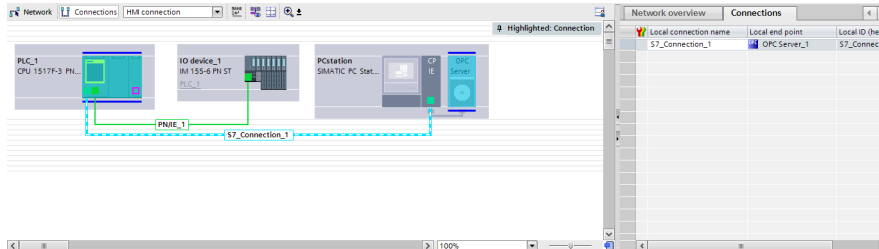


Figure 5: Final setup

If the Profinet interface connections intersects the setup is likely to fail. To solve this, check the IP-adress for each Profinet interface to ensure the I/O device and the PC station are on different subnets. If the connections still intersects, remove the I/O-device and the PC station from the configuration along with the connections. Then add the PC station to the configuration first and the I/O-device second.

Station Configuration Editor

Step 5

Open the Station Configurator Editor, see Figure 6. This may take a long time. If the Station Manager crashes or fails to load configuration, restart the computer to resolve these issues.

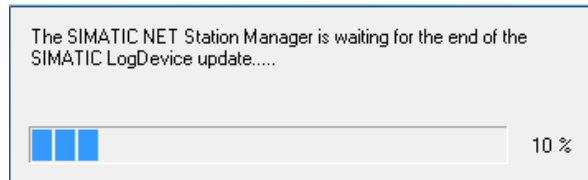


Figure 6: Station Configurator Editor start up

In the Station Configurator Editor, click "Add. . ." select the IE general in slot one, as this configuration must correspond to the one in TIA portal in order to be successful and the OPC server in slot 2, see Figure 7. If prompted, click "OK". The setup should now correspond to Figure 7.

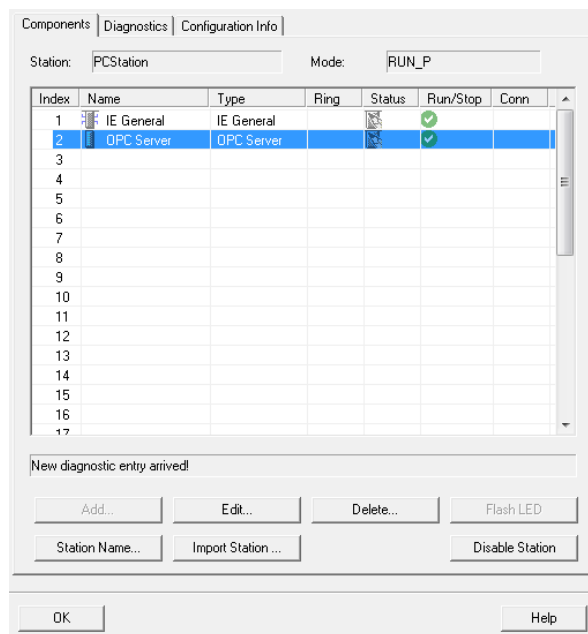


Figure 7: Station Configurator Editor

Step 6

Right click on the PC station in the Project tree in TIA portal. Select "Download to device" and "Hardware configuration". In the "Extended download to device" window that follows, enter the IP address of the IE general [11], in this guide "192.168.1.5". Click load and finish.

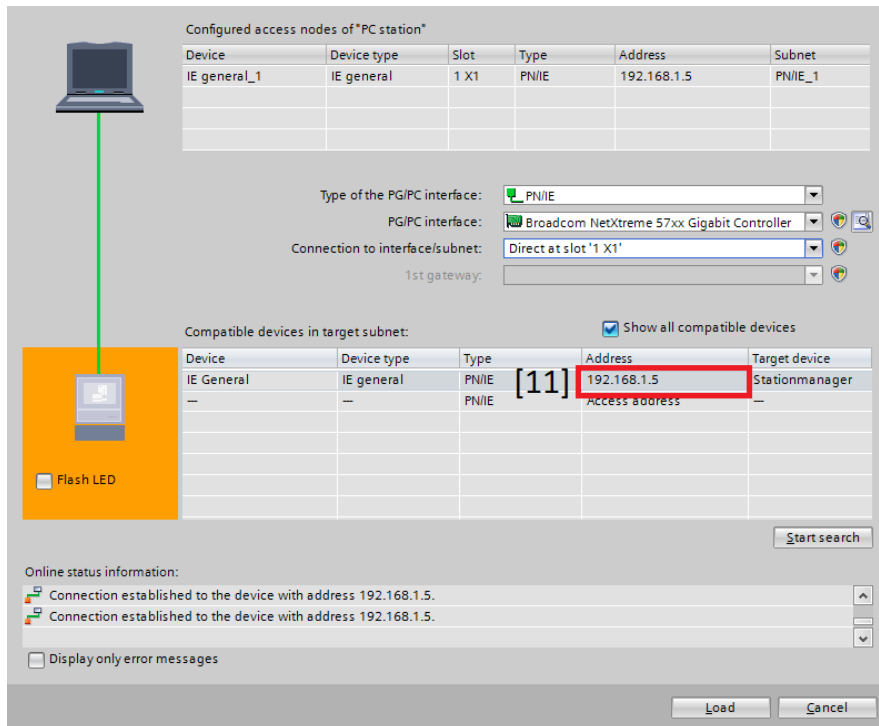


Figure 8: Extended download to device window

If faced with an error report according to Figure 9, there are irregularities in the setup of the OPC server in TIA portal and the one in Station Configurator Editor. The most common differences are either different firmware versions, or setup index orders. To check the firmware version of the components in TIA portal click on the PC station and enter device view. Right click on a device in the station and choose "Properties" [12], see Figure 10. In properties, select "General" [13] and "Catalog information" [14]. The error report should in this case state the firmware version of the Station Configurator Editor devices.

If the devices firmware does not match, right click the device in the device view and select "Change device" [15], see Figure 10. In the pop-up window simply enter a new device [16] and firmware [17], see Figure 11.

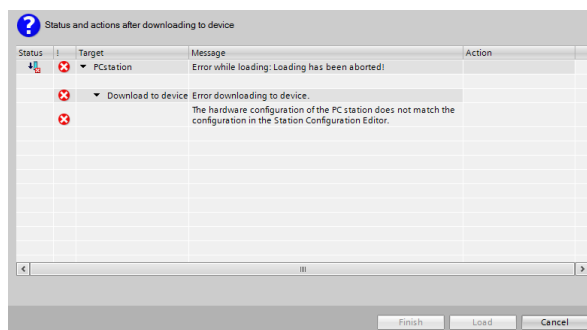


Figure 9: Error report

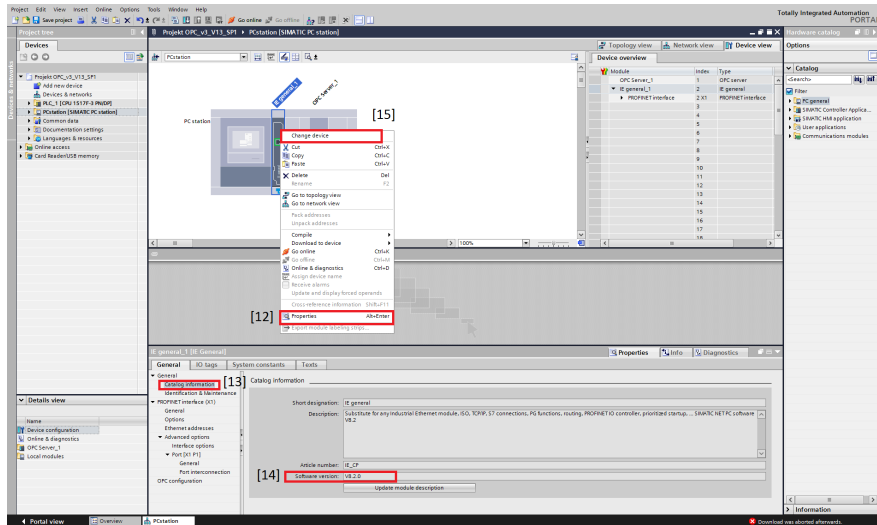


Figure 10: Accessing firmware version

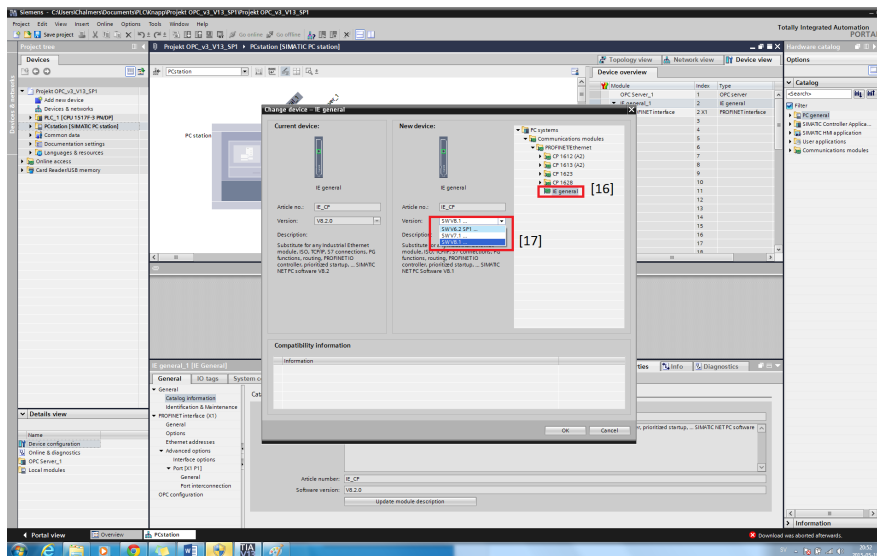


Figure 11: Changing the firmware version

F

Tutorial: OPC Tunneling With Matrikon OPC

OPC Tunneling With MatrikonOPC

May 2015

Introduction

This is a step by step guide to explain how to establish the OPC tunnel used in the project. The software MatrikonOPC Tunneller will be used.

Guide

Step 1

Download and install the Matrikon OPC Tunneller on both of the devices in the configuration.

Step 2

On both computers, add the Matrikon OPC Tunneller to the Windows firewall exception list, see Figure 1. Alternatively the firewall can be turned off. If additional antivirus software is installed on the computer these may need a similar configuration as well.

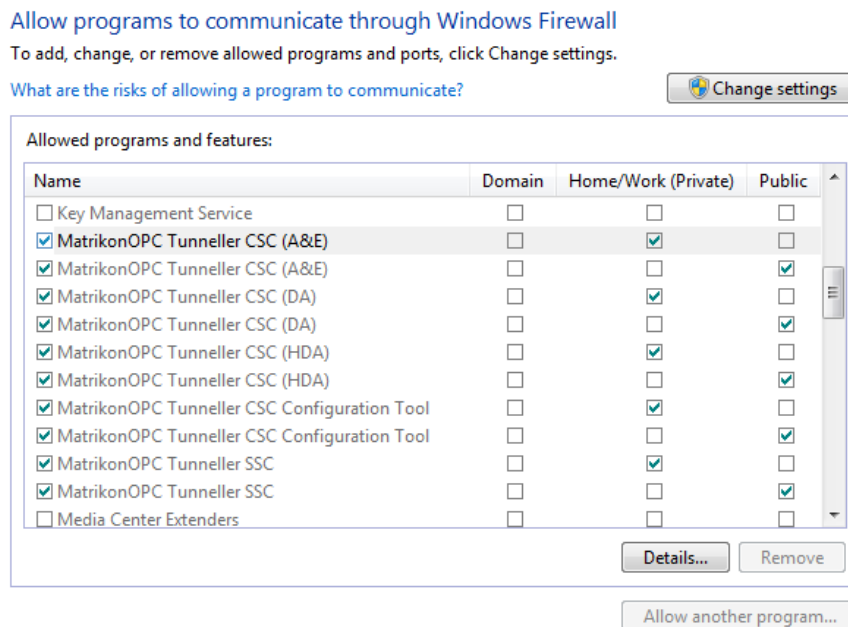


Figure 1: Windows firewall

Step 3

On the server-side computer, navigate to the installation folder, in this guide C:\Program Files (x86)\Matrikon\OPC\Tunneller\Server-Side Gateway”, and select “SSKeyManager”. Right click on the application and select, “Run as administrator” [1], see Figure 2. Ensure that the configuration corresponds to that of Figure 2.

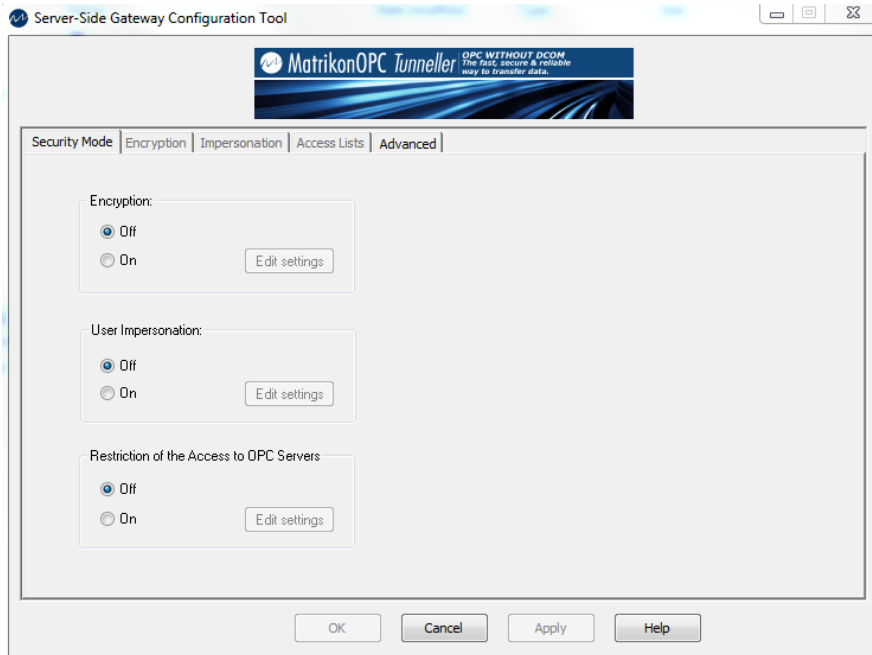


Figure 2: Server side gateway configuration

Step 4

On both computers, enter the Client-Side Gateway, right click and press “Run as administrator”. In the window that opens, press the computer icon on the top left of the screen [2], see Figure 3. This will let you configure the connection between the Matrikon client and server.

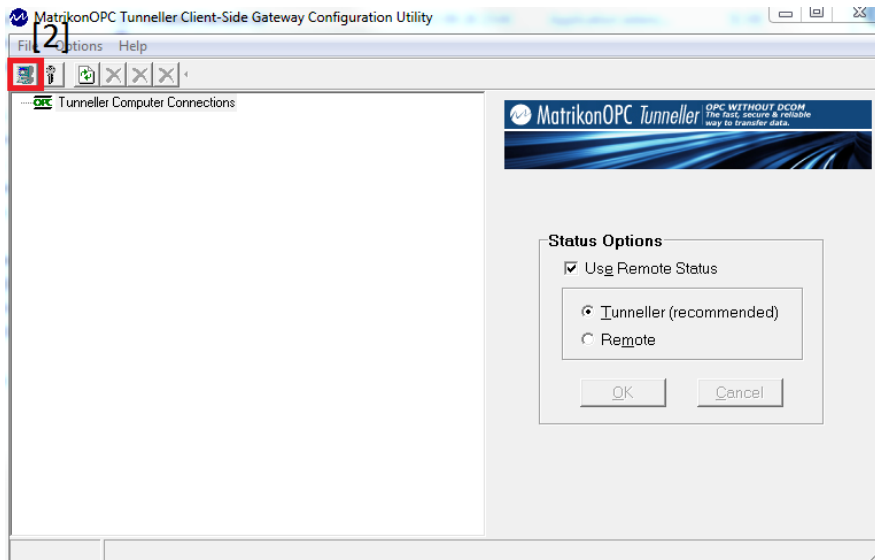


Figure 3: Client side gateway configuration

Step 5

On both computers, when the “Add Remote Tunneller Connection” window opens, confirm that the port number of the tunnel [3] is the same. Press “OK” [4], see Figure 4. It is recommended to use the standard port, in case this has to be configured, be sure no other software uses the port assigned.

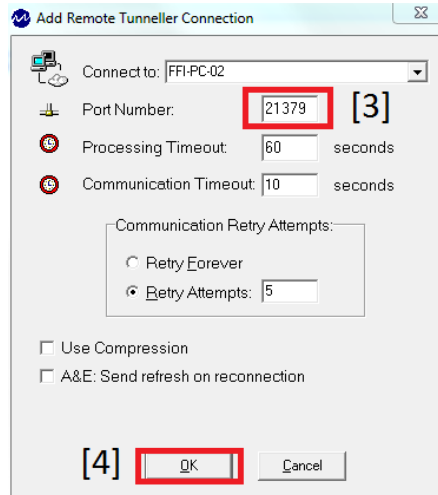


Figure 4: Port confirmation

If the configuration has been done correctly the client side of the tunnel should be as in Figure 5. The server computer is in this configuration named FFI-PC-02, its servers can be seen below [5]. The servers displayed above is simply simulations that the MatrikonOPC Tunneller can run in order to confirm the functionality of a client.

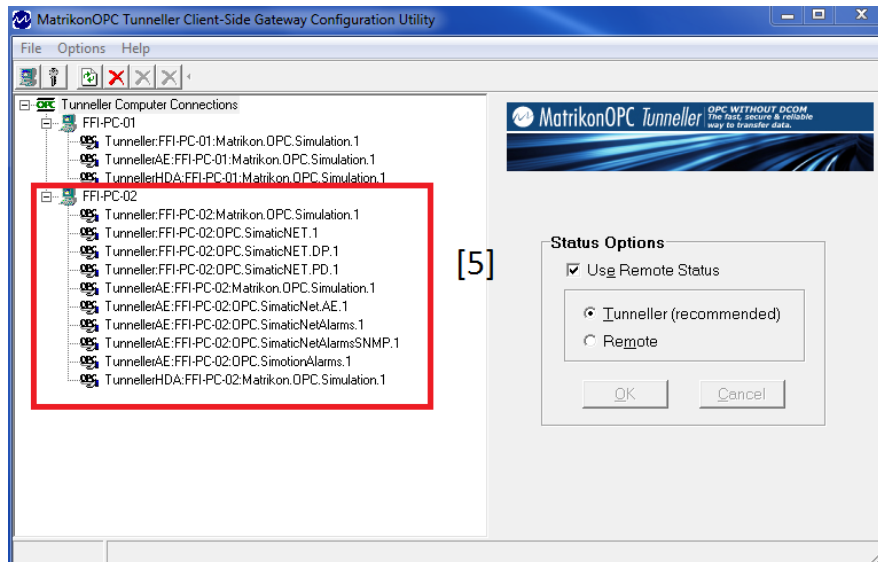


Figure 5: Available OPC servers

Step 6

To complete the setup to the client in Process Simulate, open a project. In the top toolbar, enter “Tools” [6] followed by “Options” [7], see Figure 6.

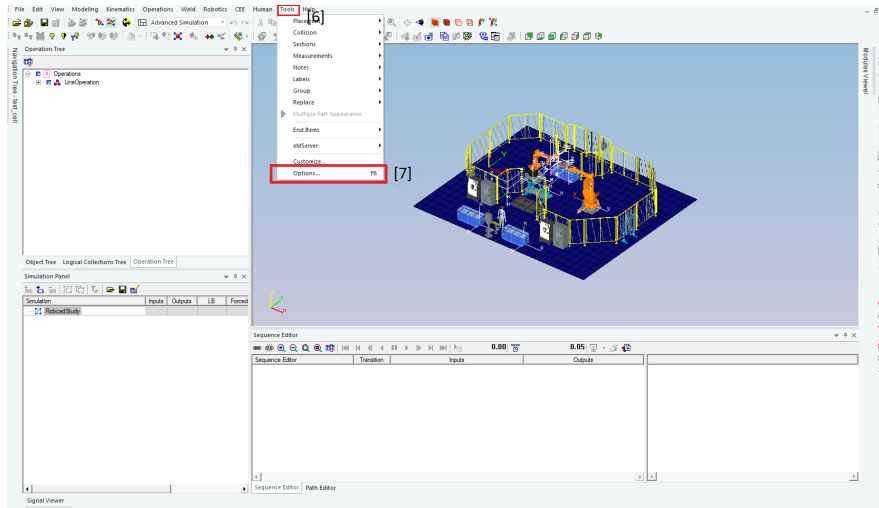


Figure 6: Process Simulate options

Step 7

Select PLC in the tabs on top of the Options window [8], Confirm “PLC” [9] and “External Connection, followed by “Connection Settings” [10], see Figure 7.

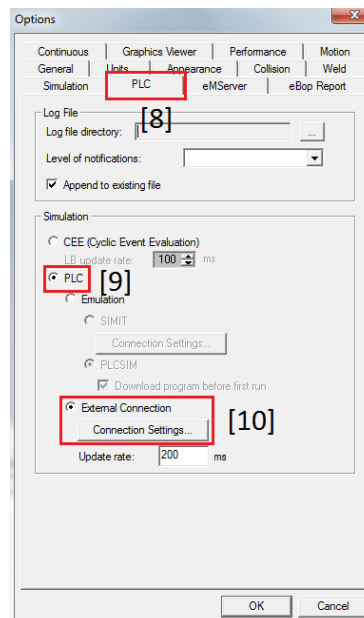


Figure 7: External PLC connection

Step 8

In the next window, see Figure 8, press “Add...” [11] in the lower right corner, and “OPC server”, in the drop down menu that follows. Press the “...” button [12] and choose the “FFI-PC-02:OPC.SimaticNET.1” [13]. Select the Connections icon in the right display [14] and press “OK” [15]. Check that the signal mapping setting is Signal names [16] and confirm with “OK”.

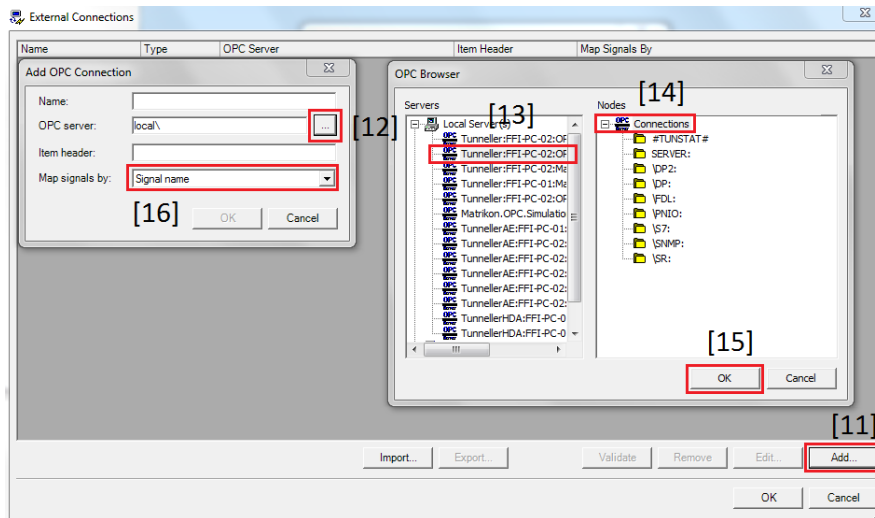


Figure 8: Choosing OPC server

Step 9

Confirm the OPC server with “OK”. The client is now connected to the server, see Figure 9.

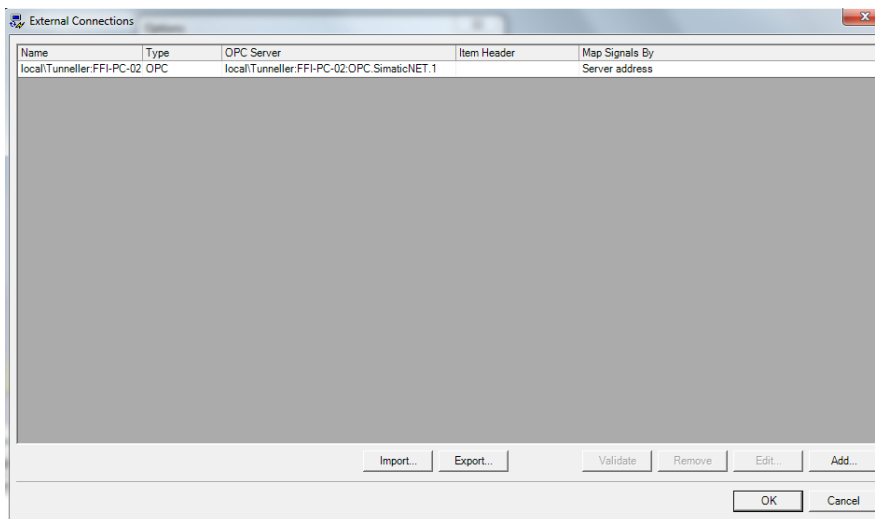


Figure 9: Added OPC server

Signal mapping

The signals of the virtual environment are controlled by variables in the OPC server. As seen in Figure 10 each variable corresponds to a variable with the same name on the OPC server.

Signal Name	Memory	Type	Address	IEC Format	PLC Connection	External Connection	Resource
Start_Button_end	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	
Clamp_Open_end	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● FixtureClamps
PART_2_to_Fixture_end	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	
PART_1_to_Fixture_end	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	
Clamp_Close_end	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● FixtureClamps
Weld_Op_end	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● abb_ir64_24_12
Clamp_Open_2_end	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● FixtureClamps
Pick_and_Phase_Op_end	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● kr30_3
Weld_Op_start	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● abb_ir64_24_12
Pick_and_Phase_Op_start	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● kr30_3
START_CYCLE	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	
Select_to_Open_Gate1	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● Gate
Gate_1_Open	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● Gate
Select_to_Open_Gate 2	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● Gate
Gate_2_Open	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● Gate
Human_SafetyMat	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● safemate
SafetyMat_On	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● safemate
Emergency_Stop_Button	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	
WeldGun_to_SDMOPEN	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● WeldGun
WeldGun_to_OPEN	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● WeldGun
WeldGun_to_CLOSE	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● WeldGun
WeldGun_at_HOME	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● WeldGun
WeldGun_to_SDMOPEN	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● WeldGun
WeldGun_at_OPEN	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● WeldGun
WeldGun_at_CLOSE	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● WeldGun
WeldGun_at_HOME	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● WeldGun
Gripper_to_HOME	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● Gripper
Gripper_to_CLOSE	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● Gripper
Gripper_to_OPEN	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● Gripper
Gripper_at_HOME	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● Gripper
Gripper_at_CLOSE	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● Gripper
Gripper_at_OPEN	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● Gripper
abb_ir64_24_12_starProgram	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● abb_ir64_24_12
abb_ir64_24_12_programNumber	☑	BYTE	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● abb_ir64_24_12
abb_ir64_24_12_emergencyStop	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● abb_ir64_24_12
abb_ir64_24_12_programPause	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● abb_ir64_24_12
abb_ir64_24_12_programEnded	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● abb_ir64_24_12
abb_ir64_24_12_mirrorProgramNumber	☑	BYTE	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● abb_ir64_24_12
abb_ir64_24_12_errorProgramNumber	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● abb_ir64_24_12
abb_ir64_24_12_robotReady	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● abb_ir64_24_12
abb_ir64_24_12_at_HOME	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● abb_ir64_24_12
kr30_3_starProgram	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● kr30_3
kr30_3_programNumber	☑	BYTE	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● kr30_3
kr30_3_emergencyStop	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● kr30_3
kr30_3_programPause	☑	BOOL	No Address	No Address	☑	localTunneller.FFI-PC-02 OPC SimaticNET.1	● kr30_3

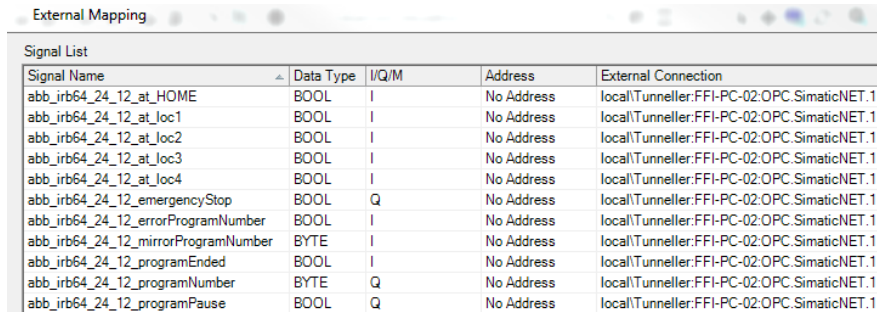
Figure 10: Signals of the virtual environment

If the variables used in the virtual environment doesn't correspond to the ones in the OPC server an error message will be displayed when running the cell, see Figure 11. In order to solve this ensure that all the variables missing are declared in TIA portal.



Figure 11: Error message

The final result will be an external mapping of all signals, a selection of which can be seen in Figure 12



The screenshot shows a window titled "External Mapping" containing a table with the following data:

Signal Name	Data Type	I/Q/M	Address	External Connection
abb_jrb64_24_12_at_HOME	BOOL	I	No Address	localTunneller:FFI-PC-02:OPC.SimaticNET.1
abb_jrb64_24_12_at_Ioc1	BOOL	I	No Address	localTunneller:FFI-PC-02:OPC.SimaticNET.1
abb_jrb64_24_12_at_Ioc2	BOOL	I	No Address	localTunneller:FFI-PC-02:OPC.SimaticNET.1
abb_jrb64_24_12_at_Ioc3	BOOL	I	No Address	localTunneller:FFI-PC-02:OPC.SimaticNET.1
abb_jrb64_24_12_at_Ioc4	BOOL	I	No Address	localTunneller:FFI-PC-02:OPC.SimaticNET.1
abb_jrb64_24_12_emergencyStop	BOOL	Q	No Address	localTunneller:FFI-PC-02:OPC.SimaticNET.1
abb_jrb64_24_12_errorProgramNumber	BOOL	I	No Address	localTunneller:FFI-PC-02:OPC.SimaticNET.1
abb_jrb64_24_12_mirrorProgramNumber	BYTE	I	No Address	localTunneller:FFI-PC-02:OPC.SimaticNET.1
abb_jrb64_24_12_programEnded	BOOL	I	No Address	localTunneller:FFI-PC-02:OPC.SimaticNET.1
abb_jrb64_24_12_programNumber	BYTE	Q	No Address	localTunneller:FFI-PC-02:OPC.SimaticNET.1
abb_jrb64_24_12_programPause	BOOL	Q	No Address	localTunneller:FFI-PC-02:OPC.SimaticNET.1

Figure 12: External signal mapping

G

Tutorial: Validating an OPC Server with OPC Scout

Validating an OPC Server with OPC Scout

May 2015

Introduction

In order to confirm the Station Configurator Editor is running properly and the OPC server is online, OPC scout was used.

Guide

Step 1

Open Station Configurator Editor and add the OPC server and the IE general, see Figure 1. Ensure that both devices are still running when the TIA portal hardware configuration has been downloaded. It is normal for the devices to shut down during this procedure, so check the status of each device after the download has finished.



Figure 1: Station Configurator Editor

Step 2

When the devices have been loaded into Station Configurator Editor, open OPC scout, see Figure 2.

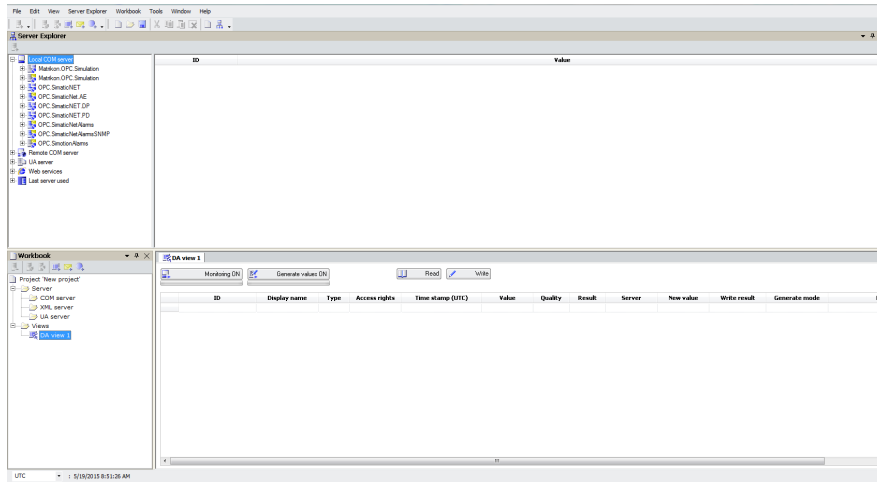


Figure 2: OPC scout

Step 3

To ensure that the server is running and detectable, select the OPC.SimaticNET in the Local COM server tree [1], see Figure 3. The detection process may take some time.

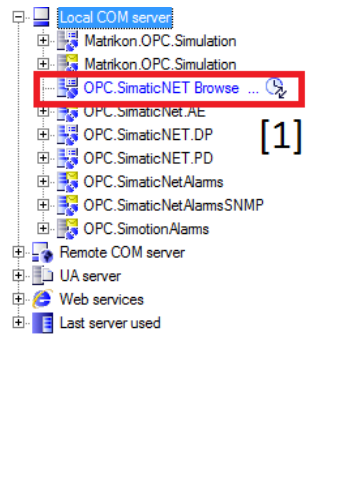


Figure 3: OPC.SimaticNET

Step 4

If the connection is successful OPC scout will read *"The server is running normally"* according to Figure 4.

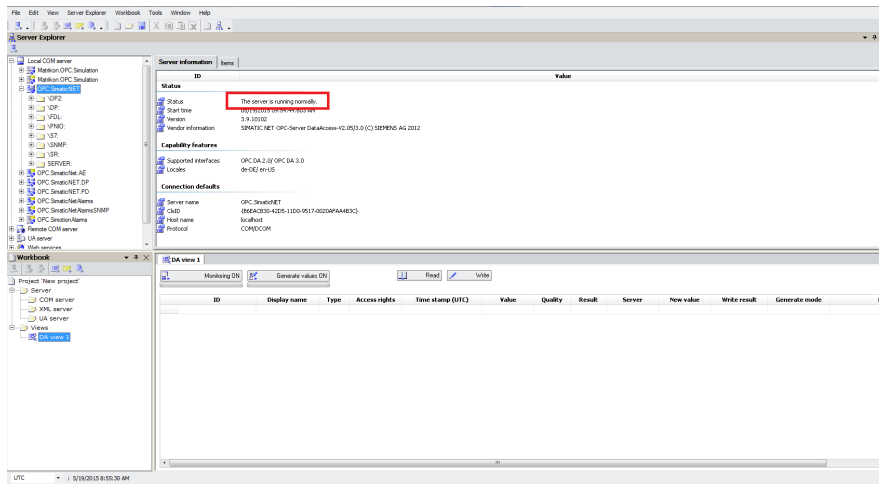


Figure 4: Status of OPC server

H

Tutorial: Writing Ladder Code in TIA Portal

Writing Ladder Code in TIA Portal

May 2015

Introduction

This tutorial will go into the basics of ladder programming in TIA portal. It will focus on where to find the different tools used when programming in TIA portal and not go into detail on how to write actual code.

Guide

Step 1

Start by opening a previous project, if none have been created, follow the "Setting up a project in TIA portal"-tutorial in order to create and configure a new one.

Step 2

In the project tree, select "Main [OB1]" [1], see Figure 1. This will open up the main block of the ladder program, this is where the code will be written.

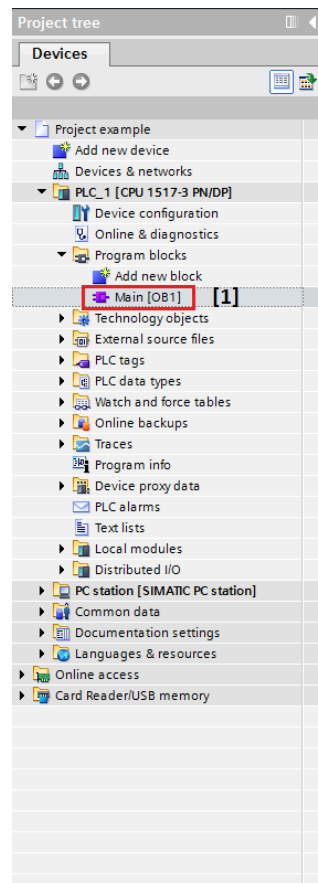


Figure 1: Project tree view

Adding logic operations

To add logic operations, drag a symbol from the menu above the networks [2] or from the instructions catalogue [3] to the right, see Figure 2. In the catalogue all available operations can be found.

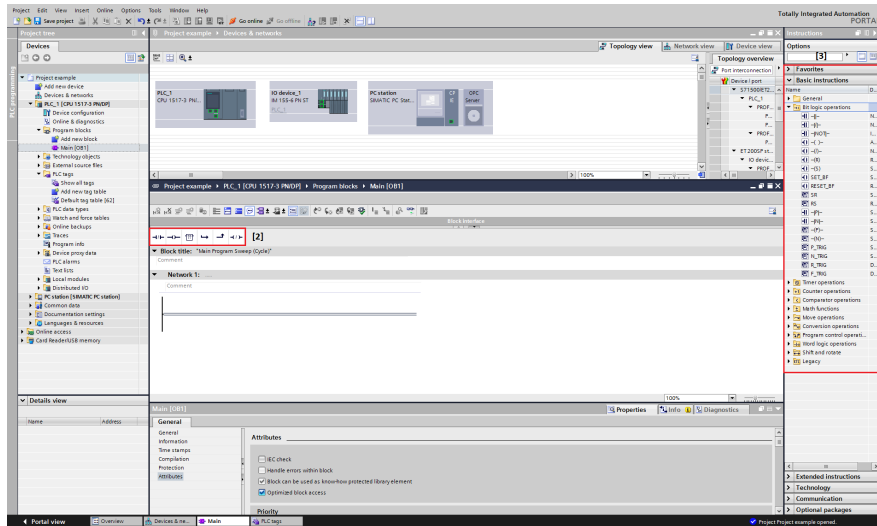


Figure 2: "Main block" view

Defining new variables

To add new variables, or tags, select "show all tags" [4] in the project menu, see Figure 3. To add a new tag press an empty field in the list [5], a new tag can be defined as an input, output or memory signal.

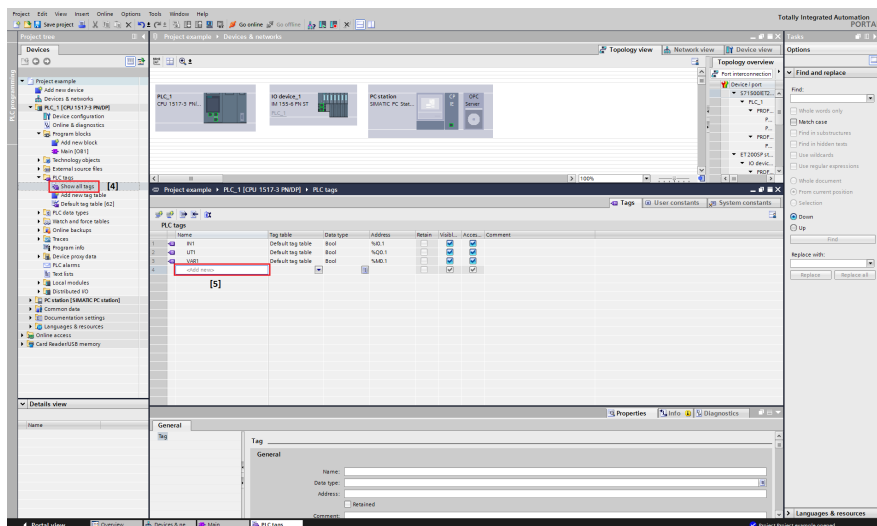


Figure 3: "Show all tags" view

This concludes the basics of how to write ladder code in TIA portal.