

DEVELOPMENT OF A RECONFIGURABLE ASSEMBLY SYSTEM WITH ENHANCED CONTROL CAPABILITIES AND VIRTUAL COMMISSIONING

By:

JOHAN NIEMANN

Thesis submitted in fulfilment of the requirements for the degree:

MASTER TECHNOLOGIAE: ENGINEERING ELECTRICAL

in the

School of Electrical and Computer Systems Engineering

of the

Faculty of Engineering, Information and Communication Technology

at the

Central University of Technology, Free State

Supervisor:

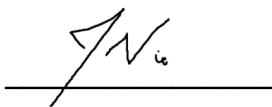
Prof. H.J. VERMAAK

Bloemfontein

2013

Declaration

I, Johan Niemann (Identity number: [REDACTED] Student number: 20438877) do hereby declare that this research project which has been submitted to the Central University of Technology for the degree MASTER TECHNOLOGIAE: ENGINEERING ELECTRICAL, is my own independent work; and complies with the Code of Academic Integrity, as well as other relevant policies, procedures, rules and regulations of the Central University of Technology; and has not been submitted before by any person in fulfilment (or partial fulfilment) of the requirements for the attainment of any qualification.

A handwritten signature in black ink, appearing to be 'JNi', written over a horizontal line.

Student Signature:

Date: 2013 – 06 – 06

Acknowledgements

The Author would like to acknowledge the following individuals and institute, whom without the completion of this dissertation would not have been possible:

- Firstly to The God Almighty, because through Him all is possible and He surely works in mysterious ways!
- The author's spouse and parents, for their love and patient support.
- Prof. HJ Vermaak, the CUT and RGEMS, for the provision of guidance, granting the opportunity to undertake the project, monetary assistance and the work experience gained during the duration of the project.
- Various fellow research students within the RGEMS research group.

Abstract

The South African (SA) manufacturing industry requires developing similar levels of sophistication and expertise in automation as its international rivals to compete for global markets. To achieve this, manufacturing plants need to be managed extremely efficiently to ensure the quality of manufactured products and these plants must also have the relevant infrastructure. Furthermore, this industry must also compensate for rapid product introduction, product changes and short product lifespan. To support this need, this industry must engage in the current trend in automation known as reconfigurable manufacturing.

The aim of the study is to develop a reconfigurable assembly system with enhanced control capabilities by utilizing virtual commissioning. In addition, this system must be capable of assembling multiple different products of a product range; reconfigure to accommodate the requirements of these products; autonomously reroute the product flow and distribute workload among assembly cells; handle erroneous products; and implement enhanced control methods. To achieve this, a literature study was done to confirm the type of components to be used, reveal design issues and what characteristics such a system must adhere to. Software named DELMIA was used to create a virtual simulation environment to verify the system and simultaneously scrutinize the methods of verification. On completion, simulations were conducted to verify software functions, device movements and operations, and the control software of the system. Based on simulation results, the physical system was built, and then verified with a multi agent system as overhead control to validate the entire system. The final results showed that the project objectives are achievable and it was also found that DELMIA is an excellent tool for system verification and will expedite the design of a system. By obtaining these results it is indicated that companies can design and verify their systems earlier through virtual commissioning. In addition, their systems will be more flexible, new products or product changes can be introduced more frequently, with minimum cost and downtime. This will enable SA manufacturing companies to be more competitive, ensure increased productivity, save time and so ensure them an advantage over their international competition.

Contents

Declaration	ii
Acknowledgements	iii
Abstract	iv
List of Figures	ix
List of Tables	xii
Acronyms and Abbreviations	xiii
Chapter 1 Introduction	1
1.1 Introduction	1
1.2 Problem Statement	2
1.3 Research Goals and Objectives	2
1.3.1 Hypothesis	2
1.3.2 Specific Objectives	2
1.4 Research Methodology	3
1.5 Layout of Dissertation	4
Chapter 2 Literature Study	5
2.1 Introduction	5
2.1.1 Reconfigurability and Flexibility	5
2.1.2 Characteristics of Reconfigurable Manufacturing Systems	6
2.1.3 Flexibility of Reconfigurable Manufacturing Systems	6
2.1.4 Design Issues Related to Reconfigurable Manufacturing Systems	7
2.2 Typical Components used within Assembly Systems	8
2.2.1 Industrial Robots	8
2.2.2 Material Transport Equipment	11
2.2.3 Actuators	13
2.2.4 Sensors	14
2.2.5 Controllers, Modules and Communications	15
2.2.6 Automatic Identification and Tracking	16
2.3 Integration of Robots and Humans in Assembly Systems	18
2.4 Industrial Safety Precautions	19
2.5 Software Platforms	24
2.5.1 Introduction	24

2.5.2	Available Software and Functionality	24
2.5.3	Software Selection	25
Chapter 3 Simulation within a Virtual Environment		26
3.1	Introduction	26
3.1.1	Dassault Systems Product Structure	26
3.1.2	CATIA	27
3.1.3	DELMIA & DELMIA Automation.....	27
3.1.4	Work Cell Hierarchy	27
3.2	The Path to Virtual Commissioning	28
3.2.1	Creating Smart Devices	29
3.2.2	Creating Control Logic	34
3.2.3	Process Plan and Simulation	36
3.2.4	Creating an Execution Environment and Virtual Commissioning	38
3.3	Solution to Conveyor Limitation in DELMIA.....	42
3.3.1	Introduction	42
3.3.2	Conveyor Geometry	42
3.3.3	Conveyor Internal Logic.....	44
3.3.4	Summery	45
3.4	Conclusion	46
Chapter 4 Methodology		47
4.1	Introduction.....	47
4.2	System Hierarchy	47
4.3	Operation and Product Flow.....	48
4.4	OPC and KEPserver	49
4.4.1	Background.....	49
4.4.2	OPC.....	50
4.4.3	KEPserver.....	50
4.4.4	KEPserver Setup	50
4.5	System Components	52
4.5.1	Material Transport System.....	53
4.5.2	Pneumatic Parallel Gripper	55
4.5.3	FESTO Cartesian Robot	56
4.5.4	KUKA KR5 Sixx R850.....	63
4.5.5	Gantry Crane	69

4.5.6	Controller Hardware.....	71
4.5.7	PanelView™ Plus 700 Human Machine Interface	73
4.6	System Software Overview	73
4.6.1	State Controller.....	74
4.6.2	Device Handler	76
4.6.3	Internal Process Controller	77
4.6.4	Multi Agent System State.....	85
4.7	Safety Precautions Implemented in the System	87
4.8	Conclusion	88
Chapter 5	Testing and Results.....	89
5.1	Introduction	89
5.1.1	System layout	90
5.2	Tests Done in Internal Process Controller Mode	91
5.2.1	Testing the Production Planner.....	91
5.2.2	Testing System Capabilities with a Process Plan	94
5.2.3	Verifying Control with a Virtual Programmable Logic Controller	96
5.2.4	Validating Logic with System Programmable Logic Controller	100
5.2.5	Analysis - Compare the Different Cell Verification Methods.....	104
5.3	Test Done in Multi Agent System Mode.....	105
5.3.1	Test Setup	105
5.3.2	Result.....	106
5.4	Conclusion	106
Chapter 6	Contributions and Conclusion	108
6.1	Introduction	108
6.2	Summary	108
6.3	Research Goals and Objectives	109
6.4	Contributions	109
6.4.1	Enhanced Control (State Machine).....	109
6.4.2	Production Planner	110
6.4.3	Device Handler (Build Algorithm).....	110
6.4.4	Simulation Model and Virtual Commissioning.....	110
6.4.5	Physical System Infrastructure	111
6.5	Future Work.....	111
6.5.1	Simulation Environment.....	111

6.5.2	Error Handling.....	112
6.5.3	Tool Reconfigurability	112
6.5.4	Rework Bay	112
6.6	Conclusion	113
	References	114
	Appendices	122
	Appendix A: Conveyor	122
	Appendix B: KUKA process simulation	122
	Appendix C: Gantry using virtual PLC	122
	Appendix D: Cartesian virtual commissioning using system PLC	122

List of Figures

Figure 1.1 Proposed physical system and flow	2
Figure 1.2 Proposed system block diagram	4
Figure 2.1 Various industrial robots [17]	9
Figure 2.2 Various end of arm tools [20]	10
Figure 2.3 Various types of conveyors [22]	12
Figure 2.4 Various industrial actuators [25]	14
Figure 2.5 Assortments of industrial sensors [27]	15
Figure 2.6 Example of a barcode [31]	17
Figure 2.7 Example of a Data Matrix [33]	17
Figure 2.8 RFID reader with tags	18
Figure 2.9 Safety products from Rockwell Automation [40]	21
Figure 2.10 Three safety risk regions [43]	22
Figure 2.11 Perimeter fencing and screening [46]	23
Figure 3.1 Product structure of Dassault Systems	27
Figure 3.2 Hierarchy of a work cell	28
Figure 3.3 Steps to design a part in CATIA	30
Figure 3.4 Assembly of parts	31
Figure 3.5 A complete assembly cell	32
Figure 3.6 Model of a cylinder in two home positions	33
Figure 3.7 Description of control allocation	34
Figure 3.8 Control block with internal SFC logic	35
Figure 3.9 Example of a PERT chart	37
Figure 3.10 Example of a Gantt chart	37
Figure 3.11 Device Control Connection Editor	38
Figure 3.12 Virtual system built in DELMIA	39
Figure 3.13 Execution environment setup window	40
Figure 3.14 OPC setup for external PLC	41
Figure 3.15 Interconnections using connection editor	41
Figure 3.16 Conveyor with resource sensors	43
Figure 3.17 SFC code implemented in conveyors	45
Figure 4.1 Network and component connection layout	48

Figure 4.2 System layout and product flow	49
Figure 4.3 OPC tags in KEPserver	51
Figure 4.4 Modifying tag properties.....	52
Figure 4.5 Conveyor system components.....	53
Figure 4.6 Tandem lift transverse conveyor	55
Figure 4.7 FESTO gripper [76].....	56
Figure 4.8 FESTO Cartesian robot	57
Figure 4.9 FESTO configuration tool software	58
Figure 4.10 Position set table (1), position profile (2).....	59
Figure 4.11 Flow of operation for motor controller	60
Figure 4.12 FESTO motor controller IO layout.....	61
Figure 4.13 Cartesian controller algorithm	62
Figure 4.14 KUKA KR5 Sixx R850 articulated robot	63
Figure 4.15 Work envelope specifications [81]	64
Figure 4.16 KUKA control panel [82].....	65
Figure 4.17 KUKA control program	68
Figure 4.18 Gantry crane	69
Figure 4.19 Gantry PLC program.....	71
Figure 4.20 Allen Bradley controller [28]	72
Figure 4.21 PanelView Plus 700 HMI	73
Figure 4.22 Software architecture	74
Figure 4.23 State machine utilized in state controller.....	76
Figure 4.24 Device handler process	77
Figure 4.25 Internal process controller state	78
Figure 4.26 Product ordering screen.....	80
Figure 4.27 Production planner	81
Figure 4.28 Twelve-position product tray	81
Figure 4.29 Example of a product string	82
Figure 4.30 Logic representation of obtaining masking strings	83
Figure 4.31 Comparing and masking scheme	84
Figure 4.32 Priority handler routine.....	85
Figure 4.33 MAS state	86
Figure 5.1 Testing to verify system	90
Figure 5.2 System layout	91

Figure 5.3 Active processes in PLC mode	91
Figure 5.4 Sequence in which products were built.....	93
Figure 5.5 KUKA robot in virtual environment.....	94
Figure 5.6 Processes present in product tree	95
Figure 5.7 PERT chart implemented in process testing	95
Figure 5.8 Gantry crane in virtual environment	97
Figure 5.9 SFC code implemented in virtual PLC	98
Figure 5.10 Interconnections between virtual PLC and gantry.....	98
Figure 5.11 Cartesian robot in virtual environment	100
Figure 5.12 Interconnections between external PLC and Cartesian	101
Figure 5.13 Matching features to be built by Cartesian.....	102
Figure 5.14 IO monitoring window	103
Figure 5.15 Active processes in MAS mode	105
Figure 5.16 PLC in slave mode.....	105

List of Tables

Table 4.1 Description of KCP Labels [82].....	65
Table 4.2 Controller features	72
Table 5.1 Simulation results for masking handler.....	92
Table 5.2 Setup for priority routine simulation	93
Table 5.3 Comparison between cell verification methods	104

Acronyms and Abbreviations

AB	Allen Bradley
AGV	Autonomous Guided Vehicle
BCD	Binary Coded Decimal
CAD	Computer-Aided Design
CATIA	Computer-Aided Three-Dimensional Interactive Application
CPU	Central Processing Unit
CRPM	Centre for Rapid Prototyping and Manufacturing
CUT	Central University of Technology, Free State
DCS	Distributed Control System
DELMIA	Digital Enterprise Lean Manufacturing Interactive Application
DoF	Degrees of Freedom
EoA	End of Arm
FBD	Function Block Diagram
FCT	FESTO Configuration Tool
HMI	Human Machine Interface
IGES	Initial Graphics Exchange Specification
IL	Internal Logic
IO	Input/output
IP	Internet Protocol
KCP	KUKA Control Panel
KRL	KUKA Robot Language
MAS	Multi Agent System
OLE	Object Linking and Embedding
OPC	OLE for Process Control
PAC	Process Automation Controller
PC	Personal Computer
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
RAS	Reconfigurable Assembly System
RFID	Radio Frequency Identification
RGEMS	Research Group for Evolvable Manufacturing Systems
RMS	Reconfigurable Manufacturing System
SA	South Africa
SFC	Sequential Function Chart
STEP	The Standard for the Exchange of Product Model Data
TCP	Tool Centre Point
USB	Universal Serial Bus

Chapter 1 Introduction

1.1 Introduction

Manufacturing companies today are faced with the increasing need for new methods of automating assembly processes that are driven by frequent unpredictable and constantly changing markets as well as the application of just-in-time production techniques, where parts are required to arrive at the work station exactly when they are needed. These market changes include the frequent introduction of new products, changes in product demand and fluctuations in batch orders [1]. The driving factor behind the design and development of flexible assembly systems is economics, with the world market demanding a wider variety of products at a constant high quality, competitively low priced, in the shortest time span, and rapid new product introduction in varying quantities [2].

The traditional inclination to resort to the use of manual assembly for complex parts, instead of rigid special-purpose lines, is mainly justified by the high cost of dedicated lines and the inability of dedicated lines to deal with any type of part variation. Dedicated lines cannot be adjusted easily to facilitate the assembly of dimensional variations of families of parts, as well as the added manipulation requirements of such variations. Manual assembly is easily justifiable in countries with low wage levels and high unemployment ratios (large availability of labour).

Any automated assembly system must compete at all levels of production to prove feasible within such a scenario (as described above). In this respect, a competitive, automated system must be capable of competitive high-volume productivity, rapid part change-over and rapid ramp-up to full production ability. Such a system must show a certain degree of flexibility with respect to the system structure and the system capabilities. A proposed system is shown in Figure 1.1. This gives rise to the concept of systems that have reconfiguration abilities or reconfigurable systems.

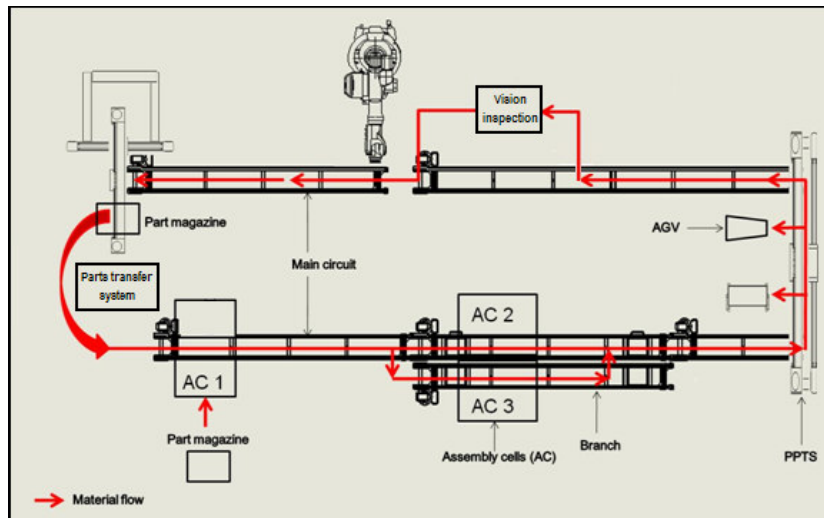


Figure 1.1 Proposed physical system and flow

1.2 Problem Statement

Dedicated manufacturing systems cannot easily be manipulated to facilitate the assembly of multiple products of a product range, and manual assembly proves too costly for the manufacturer because of high cost and/or low productivity of labourers.

1.3 Research Goals and Objectives

1.3.1 Hypothesis

South African manufacturing companies can be more competitive, increase productivity and keep expenses to a minimum by making use of reconfigurable assembly systems (RAS).

1.3.2 Specific Objectives

The objectives of this study are to:

- Develop a reconfigurable assembly system with various control options to have enhanced control capabilities.
- Develop a simulation or virtual model to verify control and operation before a real version of the system is built.

- Develop a RAS that can switch control between a Multi Agent System (MAS) and the system controller.
- Develop a system that will be able to manufacture different products of the same product family.
- Develop a RAS that is able to adapt automatically, depending on the requirements of the specific product to be made.
- Develop a RAS that is able to reroute the product flow and shift the workload between conveyors.
- Develop a system that can do error handling in case of product failure.

1.4 Research Methodology

The complexity of a control system makes it difficult to determine the performance of the system analytically [3]. For this reason, simulation software will be used to design the subsystems or assembly cells. The software choice will be DELMIA from Dassault Systems [4] [5]. DELMIA allows Programmable Logic Controller (PLC) code to be simulated in a virtual environment. This eases debugging and design faults can be rectified before the physical system is implemented.

In addition, the physical structure and design of the system will be divided into assembly cells as shown in Figure 1.2. The physical design will make use of a PLC and motion controllers for the control of the subsystems. The motion controllers will drive servo motors or stepper motors connected to linear drives. This will allow the subsystems to be reconfigured and adapt to the requirements of each product part to be made. Proximity sensors will be used to determine when and if a part has arrived at the correct position.

Additionally, the DeviceNet™ fieldbus protocol will be investigated for communications between the main controller and its peers at component level [6]. The main controller will communicate over Ethernet with the MAS to send and receive information to and from the MAS. PLC memory will be used to store information regarding the capabilities, availability and status of the subsystems equipment. This memory will allow the MAS to determine the best setup for each

subsystem depending on the product type to be made. The main controller must be able to administer full control in case the MAS system seizes to operate or if a client prefers not to have the facilities of the MAS available in the system.

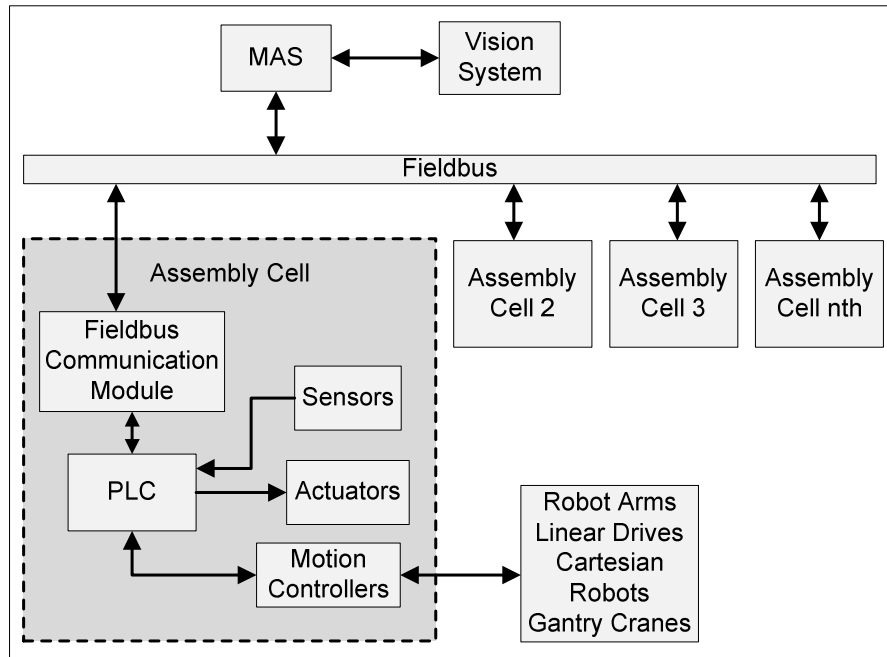


Figure 1.2 Proposed system block diagram

1.5 Layout of Dissertation

Chapter 1: This is an introductory chapter, which contains the problem statement, hypothesis and the objectives of the project.

Chapter 2: This is a literature study done by the author, to acquire preliminary knowledge in the field of study, which contains current practices in industry.

Chapter 3: This is an overview of DELMIA e.g. how the infrastructure works, how to obtain the needed geometry and control elements, and how to accomplish virtual commissioning.

Chapter 4: This is a chapter containing a description of the utilization of the proposed machinery and possible methods for implementing control elements.

Chapter 5: This is a chapter dedicated to the tests that were done, how they were done and the results obtained.

Chapter 6: A closing chapter to discuss test results as well as future work.

Chapter 2 Literature Study

2.1 Introduction

This chapter encloses a literature study done to acquire preliminary knowledge in the field of study (reconfigurable automation) and illuminate terms like reconfigurability and flexibility. In addition, the study encloses the characteristics of RMSs; provides design principles, and discusses typical components used in RMSs. Furthermore, how humans and machinery are integrated in RMSs and the safety aspects hereof are also considered. Also included in this chapter is an evaluation of available simulation software, the functionality of the software and justification for the choice of software. In brief, the chapter provides clarification on the field of study, and includes an overview of important practices currently relevant to industry.

2.1.1 Reconfigurability and Flexibility

Reconfigurability is defined by Wiendahl [7], Setchi and Lagos [8] as the operative ability to repeatedly change and rearrange the components of a system in a cost-effective way, through the addition or removal of functional elements with minimal effort and delay. The prospect of an assembly system sporting the characteristics of reusability, scalability, agility and reconfigurability has resulted in an assembly paradigm known as RASs or RMSs.

On the contrary, flexibility is defined by the tactical ability of an entire production and logistics area to switch with reasonably little time and effort to new, although similar, families of components by changing manufacturing processes, material flows and logistical functions.

Furthermore, the key differences between reconfigurability and flexibility can be elucidated by the following:

- Firstly, the diversity of work pieces handled: reconfigurable systems may switch between different families of products, while flexible systems switch between similar products.

- Secondly, the extent of change the manufacturing system has to undergo: reconfigurable systems may add or remove machine components, while flexible systems change the process or material flow.

In addition, there are two types of reconfiguration that can occur in a manufacturing system, namely basic and dynamic reconfiguration. Basic reconfiguration is reconfiguration in its simplest form, which can be achieved by stopping the system, applying the necessary hardware or software changes, and then restarting the system. This is also known as “cold starting the system”. Dynamic reconfiguration is reconfiguration which takes place while a system is still in operation, without having to stop the system [9].

2.1.2 Characteristics of Reconfigurable Manufacturing Systems

Koren et al. mentions five important characteristics of RMSs [10]. ElMaraghy summarized these and adds an additional characteristic [11]. Together these are:

- Modularity of both hardware and software components.
- Integrability for both ready integration and future introduction of new technology.
- Convertibility to allow quick changeover between products and quick system adaptability for future products.
- Diagnosability to identify the sources of quality and reliability problems quickly.
- Customization to match designed system capability and flexibility to applications.
- Scalability to incrementally change capacity rapidly and economically [11].

2.1.3 Flexibility of Reconfigurable Manufacturing Systems

There are different levels of flexibility; ElMaraghy lists ten types of flexibility as follows [11]:

- Machine: Various operations can be performed without set-up change.

- Material handling: Various paths available for transfer of materials between machines. It can be measured by a number of used paths divided by the total number of possible paths between all machines.
- Operation: Various operation plans are available for part processing. It can be measured by the number of different processing plans available for part fabrication.
- Process: Different sets of part types can be produced without major setup changes, e.g. part-mix flexibility.
- Product: Ease (in terms of time and cost) of introducing products into an existing product mix; this contributes to agility.
- Routing: It can be measured as the ratio of the number of feasible routes of all part types to the number of part types.
- Volume: The ability to vary production volume profitably within production capacity.
- Expansion: Ease (in terms of effort and cost) of augmenting capacity and/or capability, when needed, through physical changes to the system.
- Control Program: The ability of a system to run virtually uninterruptedly (e.g. during different shifts) due to the availability of intelligent machines and system control software.
- Production: It can be measured as the number of all part types that can be produced without adding major capital equipment.

2.1.4 Design Issues Related to Reconfigurable Manufacturing Systems

The degree of flexibility of a manufacturing system largely depends on its modularity. A modular design makes it easy to install, remove and regroup various modules of an assembly system. An advantage of modular design is the possibility of “plug and play/produce”. This means that modules can be dynamically added or removed from the system without having to change, reconfigure or recalibrate the hardware or software of the assembly system. Six design principles for RMSs suggested by Katz follow directly [12]:

- Design around a specific part family.

- Customized flexibility.
- Easy and rapid convertibility.
- Modular scalability, addition or removal of elements that increase productivity or efficiency.
- Allow reconfiguration so the machine may operate at several locations along the production line performing different tasks at different locations using the same basic structure.
- Should be implemented using a modular approach, common hardware and interfaces.

2.2 Typical Components used within Assembly Systems

This section discusses the typical components used in an assembly system, the importance of these components and the factors considered when choosing these components. In addition, this section shows that these components are agile, modular, diagnosable and easily assembled. Furthermore, these components are readily available because of their high demand; and are relatively cheaper than their dedicated non-flexible equivalents—due to their reusability and reconfigurability. In addition to the latter, modifications to an obsolete machine can prove more costly than purchasing a new machine. In short, this section shows the usage of typical components in assembly systems, as well as what makes them reconfigurable.

2.2.1 Industrial Robots

To quote from literature, an industrial robot can be defined as: “a reprogrammable, multifunctional manipulator designed to move materials, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks” [13] [14]. Industrial robots are extensively used in modern industry, in applications which are too dirty, dull, dangerous or difficult to be done by human beings. These applications include pick and place work, cutting, spraying, drilling, welding, assembly operations, quality inspection and heavy lifting. In addition, the choice of robot depends greatly on what environment it is used in, the required payload at a required speed, the size of the

work envelope and the accurate repeatability of motions. Furthermore, depending on the application, these robots can either be floor, wall, shelf, roof or gantry mounted to fit the application [15].

2.2.1.1 Robot Types

Moreover, these different types of robots are constructed from varying numbers of axes and axes orientations, which are linked together either by rigid, revolting or prismatic joints. Major types of industrial robots are classified by their mechanical structure and are categorized as: Articulated robots, Cartesian (also known as, gantry or XYZ robots), SCARA, Cylindrical, Parallel, Spherical or Polar robots [16]. Examples of various industrial robots are shown in Figure 2.1 below.



Figure 2.1 Various industrial robots [17]

2.2.1.2 Robot Components

Although various types of industrial robots have different mechanical structures, they still consist of the same elementary subcomponents and are distinguished as the controller, arm, drives, end of arm tooling (end effectors) and feedback sensory. These components can be described as follows [18]:

- **Controller:** Firstly, the controller in conjunction with a teach pendant, are connected to the physical robot arm and enables an operator to control, interface, program or configure the robot. In addition, the controller runs a required control program to instruct the robot to execute a sequence of movements and operations, and stipulate the velocity, acceleration and deceleration these movements must yield to.

- Arm: This is the mechanical moving part of the robot, which manipulates the location of the end effector in the working envelope. As seen in the previous section, the construction of the arm depends on the number of axes, the orientation of these axes (XYZ) and the type of robot to be constructed.
- Drives: The drives are the actuators used to cause motion to the individual axes of the arm, and are driven by either electrical motors or other types of actuators.
- End of Arm (EoA) tool: is any device or tool attached to the robot flange (wrist), which once the arm has moved to the correct position, performs a specific task. EoA tools include grippers [19], paint and welding guns, vision equipment, suction devices and EoA tool changers, and are also known as end effectors (also see actuators). Figure 2.2 shows various EoA tooling.
- Sensory: (Refer to sensors), is used to transmit feedback signals or data to the robot controller about the position of the respective axes, orientation of objects, status of the EoA tool and the surrounding environment.



Figure 2.2 Various end of arm tools [20]

2.2.2 Material Transport Equipment

There are different practices and methods concerning the transportation of raw materials, parts or products around a manufacturing facility. These methods include manual transportation by humans, the use various conveyors, chutes and slides, factory vehicles like forklifts, and autonomous guided vehicles (AGVs) or robots, which are designed to store and retrieve manufactured goods automatically. Selecting a suitable transport system depends strongly on the application at hand. Factors that need to be considered include cost-effectiveness, available workspace, size and weight of materials to be transported, safety of the method used, and the reusability if the system is to change. In conclusion, the transport system best suited for an application must be utilized by manufacturers.

2.2.2.1 Conveyors

Different types and variations of conveyors are available to industry [21], which are tailor-made to suit the individual needs of different applications. Conveyors come in a variety of shapes and sizes and it is nearly impossible to list them all. Conveyors allow for the quick, efficient and effortless transportation of materials from one point to another inside a factory. Conveyors consist of transporting belts, chains or cables which are driven mechanically, hydraulically or electrically. In addition, some conveyors only ease manual transportation (roller conveyors) or use gravity to transport material (chutes and slides).

There are a variety of important factors concerning the selection of a conveyor system. These factors include, firstly the style of conveyor that is needed (based on application). Must material be transported on a solid belt, in trays or buckets? How will the accumulation and sorting of materials be handled? Lastly, what are the size, weight and shape of the materials to be handled? Examples of these conveyor solutions are shown in Figure 2.3. The major types of conveyors include belt, chain, cart, monorail, bucket, and towline conveyors.

Following below is a list of some benefits concerning the use of conveyors:

- Conveyors are able to safely transport material point to point, where it is laborious and expensive when done manually by humans or by vehicles operated by humans (operating cost plus salaries).

- Conveyors can be installed anywhere (floor or overhead), and are much safer than using machinery like hoists and forklifts, thus increasing human safety.
- Conveyors can move materials of all shapes, sizes and weights continuously without unnecessary stoppage of the system.



Figure 2.3 Various types of conveyors [22]

In addition to conveyors transporting material, parts and pallet feeders supply the material (parts) as well as the containers (pallets) in which materials are transported. Feeders are normally situated near the start of a conveyor to feed pallets and initial parts, and near assembly cells where the parts are needed for completion of products. Contrary to feeders, sorting and orientation devices are used to route the material through the system and to ensure that material is orientated correctly so it can be picked from the conveyor. Examples of such devices include deflectors, push diverters, rake pullers, moving slats, tilt trays and cross belts. In conclusion, conveyors utilized in conjunction with auxiliary devices can enable an assembly system to operate around the clock and therefore increase productivity.

2.2.2.2 Autonomous Guided Vehicles

An AGV is a mobile vehicle or robot which makes use of sensors to navigate itself through a facility autonomously in order to achieve specific tasks. Vast varieties of AGVs are currently being implemented in industrial applications to move materials around a manufacturing facility autonomously. Examples of industrial AGVs include [23]: autonomous guided carts (AGCs), which are used similarly to conveyors, to move material around a factory floor. In addition, automatic storage and retrieval vehicles (AS/RV), which can be defined as “automated forklifts” and are used to store materials autonomously for a period of time and retrieve it for dispatch. In conclusion, the use of AGVs to transport material is just as trendy as the use of conveyors; AGVs have been confirmed to be a feasible material transport solution, but will not be considered given the scope of the project.

2.2.3 Actuators

Figure 2.4 shows examples of various industrial actuators. Actuators are devices which transform potential energy (compressed air) into actions or motion when instructed to by input signals (normally electrical signals). To rearticulate, actuators are the components which do the physical labour in an assembly system when instructed to by a controller. Moreover, common types of actuators include electrical motors, air muscles, and pneumatic or hydraulic cylinders and drives [24]. Actuators can be part of a machine, an EoA tool or an auxiliary device in an assembly cell. Furthermore, actuators are used to drive a belt or to stop, lift, tilt, clamp, push, align and orientate objects on a conveyor. In conclusion, actuators translate the instructions of the controller into physical actions.



Figure 2.4 Various industrial actuators [25]

2.2.4 Sensors

Monitoring and receiving feedback from sensors forms a vital part of an assembly system. Sensors are widely used in various and different applications. Figure 2.5 shows an assortment of different types of sensors. Basic types of sensors include sensing proximity of objects, variation in temperature, flow, as well as fluid and gas pressure [26]. The most commonly used sensors in assembly systems must be proximity sensors. It senses the presence of parts or objects, within a certain range, at a defined position. Furthermore, different types of proximity sensors are present in industrial applications. Examples of these types include capacitive sensors that sense any object in its proximity; inductive sensors that sense metal-containing objects; magnetic sensors that sense magnetic fields; and infrared (IR) sensors that sense transmitted beams of IR light, where the light can be sourced either from itself or a different source. In addition to proximity sensors, pressure and flow sensors are used to provide feedback to a controller (PLC) used in hydraulic or pneumatic systems. The feedback from the sensors is then used by the controller to determine if air pressure is present or if the system is working at or under a safe pressure. Lastly, contact switches can also be used as sensors. A common application example is using a micro-switch to detect the physical limits of a machine. In conclusion, sensors provide a system with capabilities, similarly to what senses provide to a human being.



Figure 2.5 Assortments of industrial sensors [27]

2.2.5 Controllers, Modules and Communications

A large number of industrial programmable controllers are available to the automation market, come in different types and sizes and have different capabilities [28]. The most commonly used controller in industry is the Programmable Logic Controller (PLC). PLCs include two types and can be categorized as “Brick” with a fixed number of IOs, and “rack mount” where a central processing unit (CPU) along with IO modules comprising different functions can be added into the rack. The CPU executes controller functions like scanning data and the execution of control sequences. Furthermore, aiding the CPU, IO modules which are classified as analogue inputs, analogue outputs, digital inputs, and digital outputs read data from sensors and control various actuators. In addition, these modules must comply with stringent electrical specifications and adhere to industrial standards.

An additional variation of PLC includes the Programmable Automation Controller (PAC). It has increased processing speeds, supports multiple simultaneous tasks and functions, and is used to control advanced and complex systems. In other words, it is an advanced PLC. Besides PACs, the inclusions of Distributed Control Systems (DCS) are widely used in process control. DCSs use multiple

controllers to handle particular tasks which are networked together by industrial standard communication protocols individually. Typically, instances of DCSs operate an entire factory from a central control room.

Similarly to a DCS, a PLC in conjunction with compact or field IOs can distribute control over a facility. The PLC acts as the main controller and the field IO modules provide a means of remote IOs to monitor and control. Furthermore, intelligent relays (basic PLCs) are used in smaller systems and processes. It is used instead of PLCs when the process at hand consists of basic logical functions and in cases where it is not profitable to use PLCs.

Finally, communications with the backplane of a controller can be achieved by a variety of communication protocols and fieldbuses. Standard communication protocols include RS-485, RS-232, CAN and Ethernet, where as industrial fieldbuses include PROFIBUS, PROFINET, CANopen®, Modbus®, DeviceNet™, Ethernet IP™, EtherCAT, etc. [29] [30].

In conclusion, the choice of controller depends largely on the processing speed required (scan rate), the intended scheme of control, the type and amount of expansion modules needed, the complexity and size of the system, and the finances allocated to the application.

2.2.6 Automatic Identification and Tracking

With the growing complexities of assembly systems and assembly processes, the need arises to identify and track manufactured parts or products throughout the progression of all manufacturing stages, hence traceability. Traceability provides the ability to identify and track a product during a manufacturing cycle, identify product specifications, ensure the quality of parts by eliminating human errors and collect historical data (batch numbers). Following below are typical example applications to obtain and support the traceability of products.

2.2.6.1 Barcodes and Data Matrixes

Referring to Figure 2.6, barcodes are one-dimensional, optical, machine-readable labels, which represent data by parallel lines with varying width and space between them [31]. Furthermore, a barcode reader or scanner can be used to retrieve data from the barcode, as long as line of sight to the reader is assured.



Figure 2.6 Example of a barcode [31]

In addition to barcodes, Data Matrixes represent a two-dimensional variation of barcodes [32], where data are represented by tiny squares instead of parallel lines (refer to Figure 2.7). A Data Matrix is prearranged into a “finder pattern” and a “timing pattern”. The finder pattern is normally shaped in an “L” and is used to find and orientate the symbol. The timing pattern, normally the opposite corner to the finder pattern, provides information on the number of rows and columns that are present in the symbol. The rest of the squares enclosed inside these borders contain the data concerning the product it is labelled with.

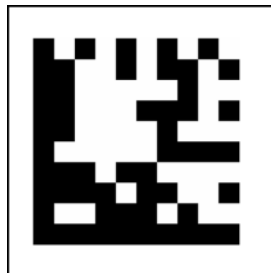


Figure 2.7 Example of a Data Matrix [33]

2.2.6.2 Radio Frequency Identification (RFID)

RFID systems consist of tags containing electronically stored information and readers that use radio frequencies to transfer identification data wirelessly from a tag to a reader on request of a querying reader [34]. Figure 2.8 shows an example of such a system. Furthermore, RFID tags can be divided into two types, namely active and passive tags. Active tags have internal batteries and are used in applications which require additional range. In contrast to active tags, passive tags have no internal batteries. Passive tags use the power induced by the electromagnetic fields received from the reader and responds by sending data back to the reader. In contrary to barcodes, RFIDs do not need to have line of sight with the reader and only require being within the specified range.



Figure 2.8 RFID reader with tags

2.2.6.3 Machine Vision

Machine or computer vision is typically used in manufacturing systems to check for quality, position, orientation and completion of products [35]. These vision systems rely on image sensors (cameras) to detect electromagnetic radiation in the form of either visible or infrared light, which is reflected by objects of interest. Furthermore, a camera is used to capture images, where filtering is applied (hardware or software filtering) and finally compared to a reference image. Further, these comparisons will determine what manipulations must be done to the object in the region of interest. However, machine vision is a broad field and will not be discussed in depth.

2.3 Integration of Robots and Humans in Assembly Systems

Assembly systems can consist of either robots or humans or a combination of both. The integration of robots and humans must be planned in such a way that the system operates safely and efficiently. However, this may increase the need for human detecting sensory, making the system more complex and increasing the chances of system failures.

In addition to increased sensory, a complication may arise which is known as manumation [36]. Manumation is the failure of implemented automation systems (hardware and software) to automate a manual work process. Furthermore, manumation may require more human involvement than before the implementation of automation components and therefore fails to achieve automation. Thus, it is important to design a system with no redundant

components present in the system. The next paragraph lists applications where robots are preferred to humans.

Typically, industrial robots are used instead of humans in difficult, dangerous and repetitive applications [37]. A robot can lift heavy objects; perform operations with great precision, repeatability and speed; save labour cost; and never shows fatigue or makes errors associated with fatigue. Furthermore, robots are used in cases where a product is harmful to a human (radioactive material), and where the presence of a human can contaminate a product (pharmaceuticals) [38].

In contrast, some applications necessitate the use of humans instead of robots. These applications include processes which are too delicate and have too many variations for the utilization of a robot. In addition, some applications need human interpretation and occasionally humans have better speed and quality. In additional cases, humans are favoured where it is not profitable to use robots; where robot systems are too complex to maintain and where a shortage of skilled programmers exist.

It is ideal to have a fully-automated assembly system, but this section has shown that certain applications have specific needs and it can be beneficial to utilize both robots and humans.

2.4 Industrial Safety Precautions

Safety measures must unquestionably be the most important issue to consider in industry. No monetary value can weigh up against a human life or a serious accident. In addition to human safety, safeguards must be taken to prevent collisions between system components and self-inflicted system component damage.

This presents two conflicting points of view concerning industrial machinery and robots. The utilization of industrial robots can increase the safety of humans, where robots instead of humans are implemented in hazardous environments and unsafe working conditions. Contrary to securing human safety, the presence of industrial robots can sometimes be the cause of an environment being hazardous. Therefore, it is crucial that manufacturers have safeguards in place to eliminate unforeseen potential hazards and ensure the safety of humans.

2.4.1.1 The E-Stop

The most important safety precaution, which is universally used in assembly systems in industry, must be the emergency stop (E-stop) [39]. During an emergency, the E-stop is used to stop the system as quickly and safely as possible. The E-stop must be easily accessible; recognizable; must work safely and reliably; and always be used as a last resort. It may never be a push button or be part of control logic (PLC program). It can be a grab wire, hand-held pressure switch, an unenclosed foot pedal or a combination of the mentioned devices. The location must be obvious. It can be in proximity of a machine, adjacent to a work cell or in the vicinity of a supervisor's post. Furthermore, the E-stop should be a red mushroom shape (top left corner, Figure 2.9), mechanically latching switch, preferably on a yellow or orange background. When the switch is actuated, it should break the continuity of the contacts (open circuit), removing the power to final power relay, actuating the power brake system using hardware-based components, and compel the system to a safe state. When resetting the E-stop, the system must remain in a safe idle state, preventing machines from restarting until the system returns to safe operating conditions.

With the use of an E-stop system, the following questions could be raised:

- Is it safe to have the E-stop system interrupt the power to motor drives and actuators?
- Should the E-stop system actuate an emergency brake or clamp?

Disconnecting the power to devices may result in hazardous “freefalling”, leading to a more disastrous, unpredictable and dangerous situation, which might cause machine damage or injury. In contrast to interrupting the power supply, considering the use of self-braking motors, which applies the brakes during a power failure or when an E-stop has been actuated, would increase the safety of the situation. It should be determined whether stopping a machine in position would worsen the situation by increasing the severity of an injury or leave the system in the safest possible state. Some power tools (power drill) used as EoA tools, still take a period of time to come to a complete standstill. In this case it would be recommended to apply the emergency brakes. Another consideration would be to allow the machine to operate in a reverse direction to a safe position.

If the EoA tool poses no threat, reversing the machine movements may free an operator from being trapped between the machine and a fixture.



Figure 2.9 Safety products from Rockwell Automation [40]

2.4.1.2 Potential Hazards

The potential hazards can be divided into the following three categories [41] [42]:

Firstly, “impact hazards” which include scenarios where a human is being struck by a moving robot arm, EoA tool or a carried part. In addition, the robot can drop or fling (throw) work pieces and parts, which also result in injury due to impact. In addition to impact hazards, “trapping hazards” include scenarios where a human is being caught between a robot arm and fixed features or perimeter fencing, resulting in being crushed. Trapping also include when a limb gets caught inside the mechanisms of a machine. On the contrary, health hazards which exclude impact and trapping are specified as “other hazards”. This is when the application, in which the robot is used, produces by-products and as a result exposes the perimeter to hazardous elements. Examples of such hazards include exposure to ultraviolet rays and sparks during welding, harmful vapours from spray painting and high levels of noise from the surroundings. Health and safety issues can be resolved by proper implementation and installation of preventative safety measures.

2.4.1.3 Preventative Measures

Alongside E-stop procedures, it is recommended that the hazardous conditions discussed in the previous section rather be prevented by well-planned installation of safeguards instead of allowing potentially dangerous situations. The level of danger can be determined by the probability of an accident occurring, and by identifying potential exposure to harmful elements. These risks can be classified into three levels; level 1 being the lowest risk of danger and level 3 being the highest [43]. By referring to Figure 2.10, the following are examples of hazardous areas: the outer limits of a work cell (level 1), where the system components pose no great threat to anything outside this border; inside the envelope of a work cell (level 2); and in close proximity of the working areas of machinery and robots (level 3).

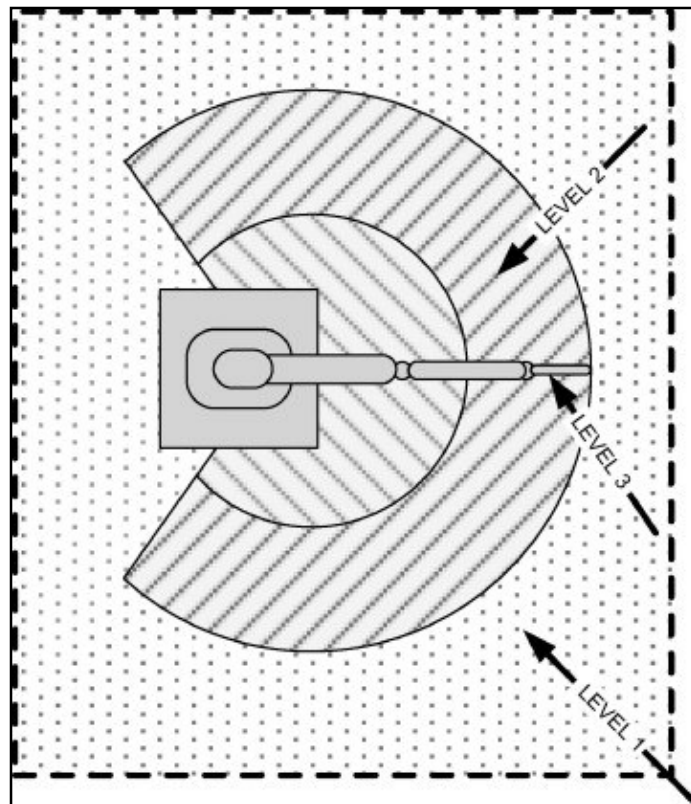


Figure 2.10 Three safety risk regions [43]

To maintain a low risk of danger in a system, preventative safety measures must be considered and implemented. Following below is a discussion on the topic [44] [45]:

Firstly, by providing factory staff with the necessary training in regards to safety procedures (safety rules), publicizing warning signs in proximity to potential hazards, and enforcing the use of protective clothing (hardhats) would prepare staff members in evading the threats present in their surroundings.

Secondly, referring to Figure 2.11, prevent staff members physically from entering an unsafe area through the installation of perimeter fencing. This can be a physical cage, structure or enclosure. It can be implemented with interlocking systems, where a robot ceases to operate when the cage gate is opened. Furthermore, securing of the perimeter by installing screening between work areas also prevents environmental and impact hazards like exposure to ultraviolet rays and the robot flinging work pieces respectively. Enforcing physical perimeter fencing will ensure that the system maintains a level 1 safety risk. In addition to physical perimeter fencing, infrared light curtains (top right corner, Figure 2.9) and perimeter beams can be installed. Curtains and beams form a virtual boundary and provide feedback to robot controllers once the light beams are interrupted. However, the use of curtains without screening off machines and robots will decrease safety if the possibilities of impact hazards are not addressed.



Figure 2.11 Perimeter fencing and screening [46]

Lastly, if personnel must work inside the border of the work cell or near the working envelope of a robot, emphasis must be placed on reliable sensing of

human proximity. The function of such sensing equipment is to prevent machinery from injuring personnel working nearby. Safety mats (bottom right corner, Figure 2.9) or pressure sensing mats are placed in a safe area close to machinery to ensure safety when programming and calibrating the machine. Further, safety solutions like SafeZone™ [47] from Rockwell Automation (bottom left corner, Figure 2.9), emit dispersed infrared light and use the reflection to sense the invasion of humans or unwanted objects within a pre-configured safety zone.

To conclude, it is necessary for manufacturers to implement safety equipment to ensure the safety of their employees, as well as protection against damage of their factory equipment.

2.5 Software Platforms

2.5.1 Introduction

There are huge varieties of software available to a large and diverse industry. These industries include the automotive, aerospace, shipbuilding, construction, machining, welding, cutting, spraying—and the list continues. For the scope of this thesis, focus will be exclusively on Product Lifecycle Management (PLM) software for automation, where PLM software is used to manage the entire lifecycle of a product. In other words, PLM software supports the user from the product as a concept design to the product in service.

2.5.2 Available Software and Functionality

Choosing the right software can be a tedious task. Additionally, the software choice must accommodate the complexity of the consumer's needs; include full technical support from the vendor, and at an affordable cost of ownership.

When browsing the internet for “computer-aided” software, multiple vendors come to the forefront offering world-class software solutions. The leading software solutions used by companies in industry, shown by software reviews and articles, include DELMIA [4] [5], Siemens PLM [48], WinMOD® [49], 3D-Automate [50], Virtual Universe [51] and other smaller packages. Incidentally, the above-

mentioned software solutions promise the same type of functionalities and similarities.

Moreover, functionality concerning geometry includes design and drafting, importing and supporting various file formats, digital mock-up and modelling. Furthermore, vendors promise planning functionalities like factory and resource layout, assembly process planning, bottleneck detection, crash and stress analysis. Additionally, internal control logic and device tasks can be allocated to imitate the operation of a system. Using the virtual version of a system, a PLC can be connected via a desired fieldbus and then validated to speed up the real commissioning and optimizing of a system.

It is clear that the functionalities of the abovementioned software solutions enables a consumer to be more productive, more competitive, save time and ensure a good product. The choice strongly depends on whether a vendor's product will suffice in terms of the consumer's needs.

2.5.3 Software Selection

As a result, WinMOD, 3D-automate and Virtual Universe will be excluded from the study due to vague descriptions in the vendor's webpage and limited access to information about its capabilities. This leaves DELMIA and Siemens PLM as the best viable solutions [52] [53]. The fact that DELMIA offers a more complete solution, has an explanatory online help archive with sample projects, is readily available to the South African market, and by contradicting favouritism to a specific PLC vendor (multiple different PLC products from different vendors are used in this project), made DELMIA an apparent choice. The only drawback of this choice is that DELMIA requires a steep initial learning curve, but this applies to Siemens PLM as well. In conclusion, DELMIA provides the recommended all-inclusive software suite that is needed to model and verify the system under development.

Chapter 3 Simulation within a Virtual Environment

3.1 Introduction

This chapter provides the specifics of virtual commissioning such as, what is virtual commissioning? What is the significance of it? How is it implemented in industry? What path is followed to obtain it? By definition, virtual commissioning is the commissioning of an assembly system within a virtual environment without needing to develop a physical system beforehand. By obtaining virtual commissioning, design flaws can be rectified early in the design stage; space reservation can be allocated for the machinery used in the system; and controller software can be verified well in advance, before building the physical system. Virtual commissioning also allows for easy reconfiguration of an existing system, where process, software or hardware changes can be made in the digital model of the system, then analysed to see how these changes influence the system, and then—based on the results, the physical system can be modified, preventing costly downtime of the physical system. Furthermore, in industry it enables manufacturers to streamline an assembly line, where planning is done more efficiently and through-put can be predicted due to the visualisation of the assembly line. In short, virtual commissioning is established when a virtual factory is controlled by either a virtual system controller (virtual machine) or a physical system controller (PLC) to emulate the behaviour of the physical system.

3.1.1 Dassault Systems Product Structure

Dassault Systems, also known as the 3D Experience Company, provides industry with a virtual universe to envision sustainable innovations. Furthermore, Dassault Systems offers a wide collection of products, but for the purpose of this study only CATIA, DELMIA and DELMIA Automation will be evaluated [54].

Figure 3.1 represents an overview of Dassault Systems' product structure and how the assorted programs interrelate. CATIA, DELMIA and DELMIA Automation are interconnected, where each product has distinctive functions, like geometry creation, simulation and analysis to attain virtual manufacturing, but all the respective functions are available in one workbench. Together, CATIA, DELMIA

and DELMIA Automation, along with other additional products, form Dassault Systems' Product Lifecycle Management (PLM) software suite.

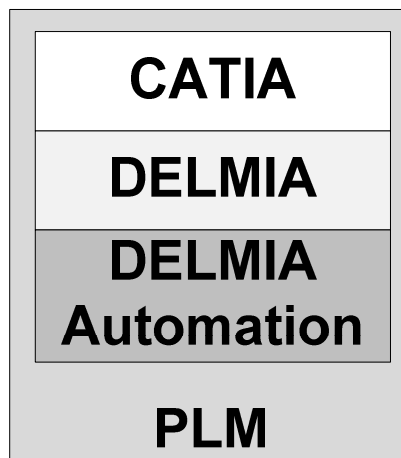


Figure 3.1 Product structure of Dassault Systems

3.1.2 CATIA

This is Dassault System's initial product (Computer-Aided Three-dimensional Interactive Application). It addresses the entire lifecycle of product development, from product concept specification through design, analysis, and simulation to final product in service [55].

3.1.3 DELMIA & DELMIA Automation

DELMIA is the PLM digital manufacturing software from Dassault Systems. It allows manufacturers to build a virtual version of a production facility and accompanying equipment. It handles early process planning through monitoring and control to final commissioning of the system [5].

3.1.4 Work Cell Hierarchy

Before continuing with the subsequent sections, it is important to know how "Products" are structured within the DELMIA environment. DELMIA uses a tree structure to build up geometry, where a "Product", is the root element of the hierarchy and contains multiple subelements or parts to represent the branches of the tree. Figure 3.2 clearly shows that a collection of parts are grouped together to form an assembly. Internal logic written in SFC language is then

allocated to these assemblies to form what is called “smart devices”. Similarly, multiple smart devices can be used together to form work cells. Basically, Figure 3.2 demonstrates how a complex system can be built up from multiple levels of subassemblies or smaller parts.

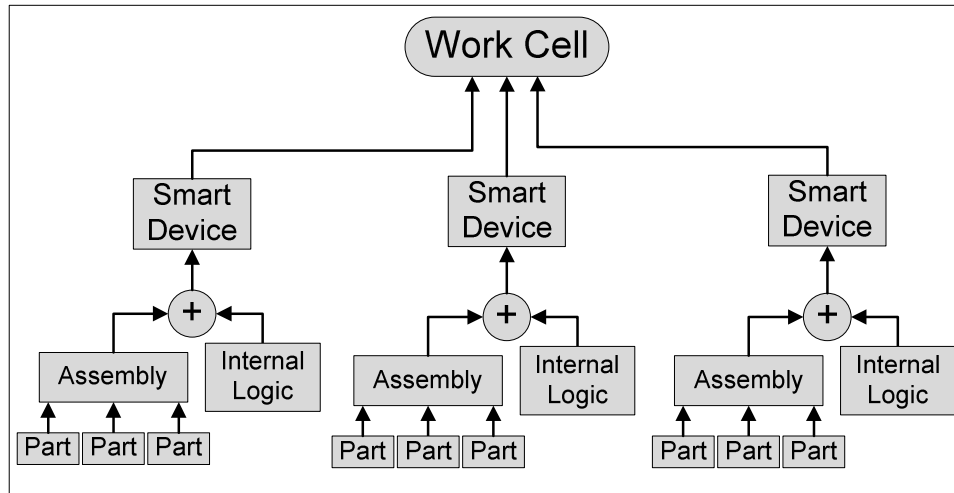


Figure 3.2 Hierarchy of a work cell

3.2 The Path to Virtual Commissioning

To obtain virtual commissioning, some preparations must be done ahead of time. These preparations include that the user must acquire prior knowledge of the software environment under discussion. Furthermore, the proposed devices or machines to be used in the system and the intended layout for these must be known. Additionally, knowledge of the behaviour and kinematics of these devices should be known, as well as how the multiple devices are to be interconnected. Then the geometry used to represent these devices, which can either be downloaded from a vendor’s website if it is available or must be designed from start, must be acquired. Thereafter, the geometry must be assembled into smart devices which entail the allocation of kinematic data, as well as internal logic behaviour. Then, ultimately, the control software utilized to operate the various devices must be developed.

In addition, all the subsequent methods utilized to realize virtual commissioning are obtained from the online help documentation which are installed as part of the software suite, but are also available online [56] [57].

3.2.1 Creating Smart Devices

As described in section 3.1.4, smart devices are built up using parts, assemblies and internal logic. A smart device can be anything from a basic linear drive or an entire multi axis machine. Fundamentally, a smart device is simply geometry with intelligence. The succeeding sections describe how to acquire the building blocks needed to obtain smart devices which are then later used for virtual commissioning.

3.2.1.1 Creating Parts

Parts are the most fundamental elements of any geometry and there are two methods of attaining these parts. Downloading the parts from a vendors' website is one method or, alternatively, parts can be designed and created using CATIA. Important to know is that multiple types of file formats can be imported into the CATIA environment. File formats like "CATpart" and "CATproduct" are native to CATIA and are represented in a tree-type structure when created. Further, this means that the parts are modifiable and can contain other smaller parts within the structure of a part. On the contrary, universal formats like "IGES" (Initial Graphics Exchange Specification) and "STEP" (The Standard for the Exchange of Product Model Data) are compatible with various other popular computer-aided design (CAD) software packages and can be imported into CATIA as well. These formats are simply graphical representations of a part, which means that their geometry cannot be modified, thus are of lower quality and use less computer-memory during generation.

Informatively, Figure 3.3 shows the typical steps taken to create new parts [58]. Number 1 in the figure shows how geometry is initially sketched, starting with a rough profile of the part. Afterwards (shown at numbers 2 and 3), the dimensions and constraints are defined, which includes constraints like distance, offset, parallelism, perpendicular and radius. In turn (shown at number 4), the sketch is transformed into a "solid" using the "pad" tool, whereafter further detailing to the part (solid) can be done by adding features like holes, pockets, grooves and shafts (shown at number 5 and 6). Additionally, after the parts are completed, it can be rendered, which means that the colour and material type can be specified. Then finally, the created parts are saved in an archive or catalogue for future

reuse within other parts or assemblies. To conclude, this section captures the importance of being able to obtain parts, through either downloading or creation.

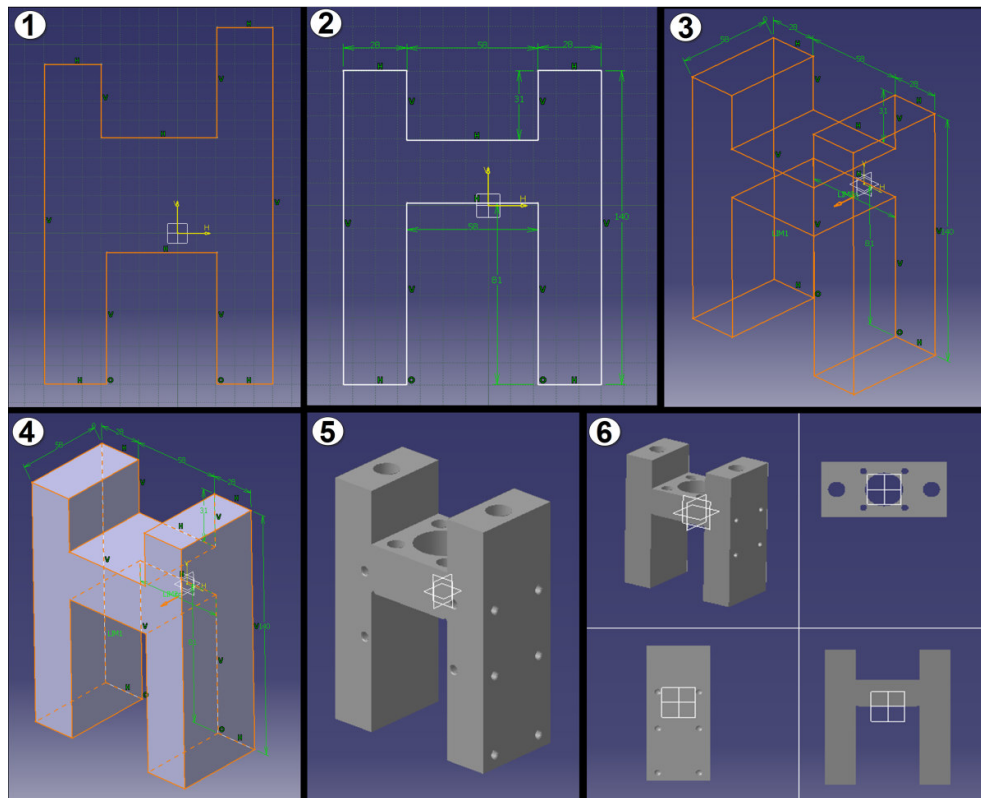


Figure 3.3 Steps to design a part in CATIA

3.2.1.2 Creating Assemblies, Mechanisms and Tasks

An assembly is a collection of parts, linked together by means of specified constraints. In order to understand assemblies, it is important to distinguish assemblies into two groups, namely rigid and moveable assemblies. Firstly, rigid or static assemblies imply that parts are assembled together into one unyielding unit. An example is using multiple small parts to construct one fixed, bigger part. On the contrary, moveable or alterable assemblies represent a mechanical assortment, which contain at least one fixed part and other moving parts. An example of a moveable assembly is a piston driven by a crank. In DELMIA terminology, alterable assemblies can also be referred to as “mechanisms”, where mechanisms are a collection of numerous “joints”.

The steps taken to construct these assemblies and subassemblies can be better explained by referring to Figure 3.4 [59]. Firstly, the parts must be imported into the environment (shown at number 1). Next, the parts must be assigned constraints, which include alignment, orientation of parts, surface contacts, offsets and fixed-part constraints (shown at number 2 and 3). On completion of constraint allocation, the geometry can be updated, causing the parts to rearrange into their intended positions (shown at number 4). This is a typical example of a subassembly or a component.

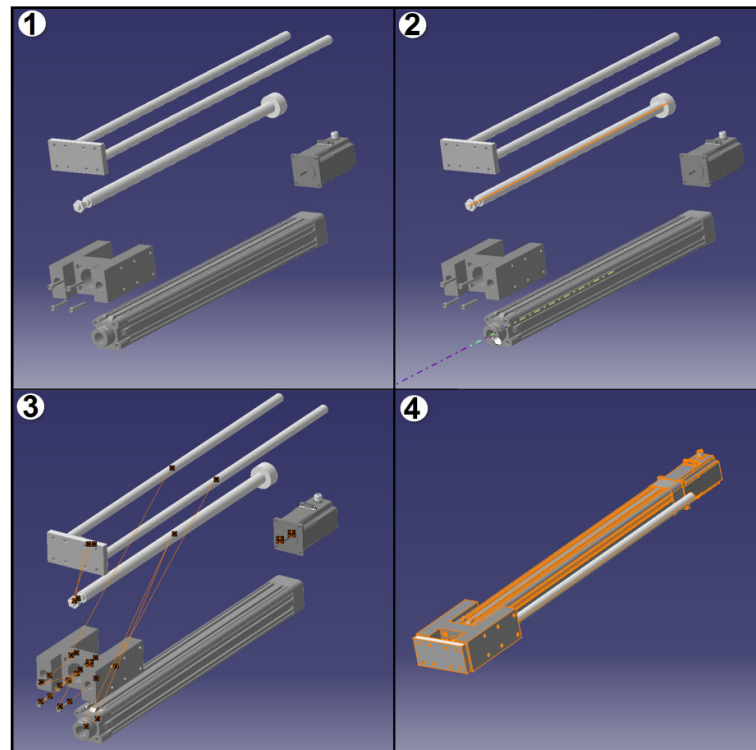


Figure 3.4 Assembly of parts

Afterwards, several joints, which represent the most basic form of moveable assemblies, are created. Joints can either be created manually or the constraints allocated to the assemblies can be converted into joints automatically. In addition, joints can be divided into several different types namely prismatic, rigid, revolute, cylindrical, and other joints. Furthermore, depending on the type of joints created, kinematic commands and relations are specified to define the degrees of freedom (DoF) of a mechanism [60]. Primarily this involves the specification of physical

limits, direction of movement, speed, and acceleration properties of the joints. This is followed to create the joints of an entire machine.

In a similar fashion to joint creation, different components can be assembled together to form a larger, more complex assembly [60]. Figure 3.5 shows how the components are first imported (shown at number 1), then aligned into position (shown at number 2) and finally attached. These attachments can be seen in the figure at number 3. The components are attached in a parent-child manner (shown at number 3), which causes the components to move relative to others. The example in the figure shows that the Y-axis (child) must move as if it is fastened to the slider of the X-axis (parent). Likewise, the cylinder (child) moves with the Y-axis (parent) and the gripper tool (child) moves with the cylinder (parent) and so forth.

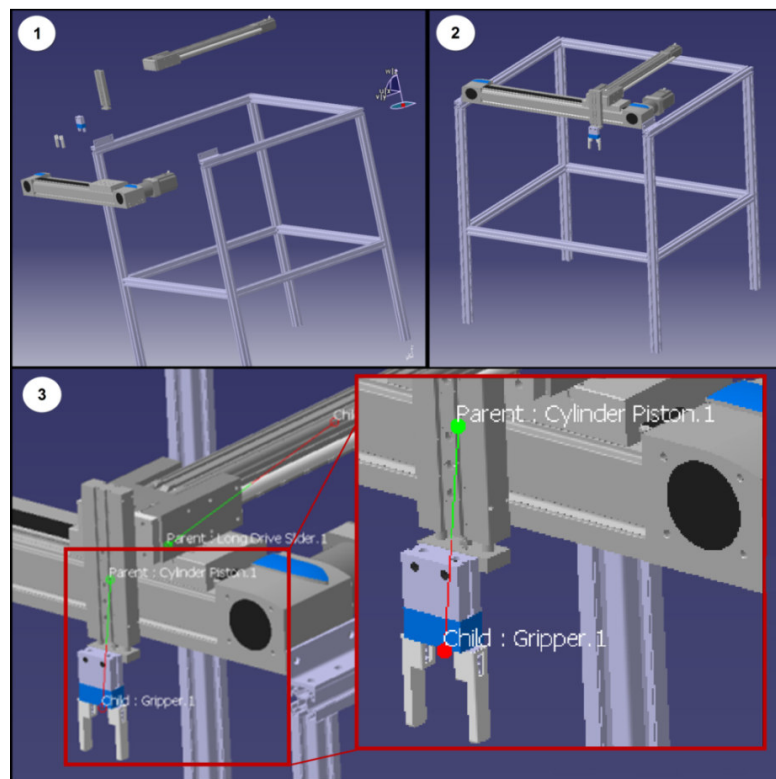


Figure 3.5 A complete assembly cell

On completion, a sequence of possible activities can be identified to demonstrate the performance of the mechanisms. These activities can be grouped into tasks and operations. To differentiate, an operation depicts the movement of a joint or

several joints when executed, whereas a task executes a series of consecutive operations and functions. The steps followed to create these tasks are described below [61].

Figure 3.6 shows a cylinder in two different, pre-taught home positions. Home positions are taught either at the absolute limits or at regular reoccurring positions on the device e.g. up and down. Teaching is done by jogging the device into position (using the jog panel), and then saving the position with a descriptive name. The taught positions are stored in a program inside the device hierarchy along with other functions like, grab, release, delay and can then be called (like a subroutine) to examine the task behaviour. At this stage, the device geometry and kinematic behaviour are created and defined completely. This enables the user to finally apply logic behaviour and use the device geometry as a smart device.

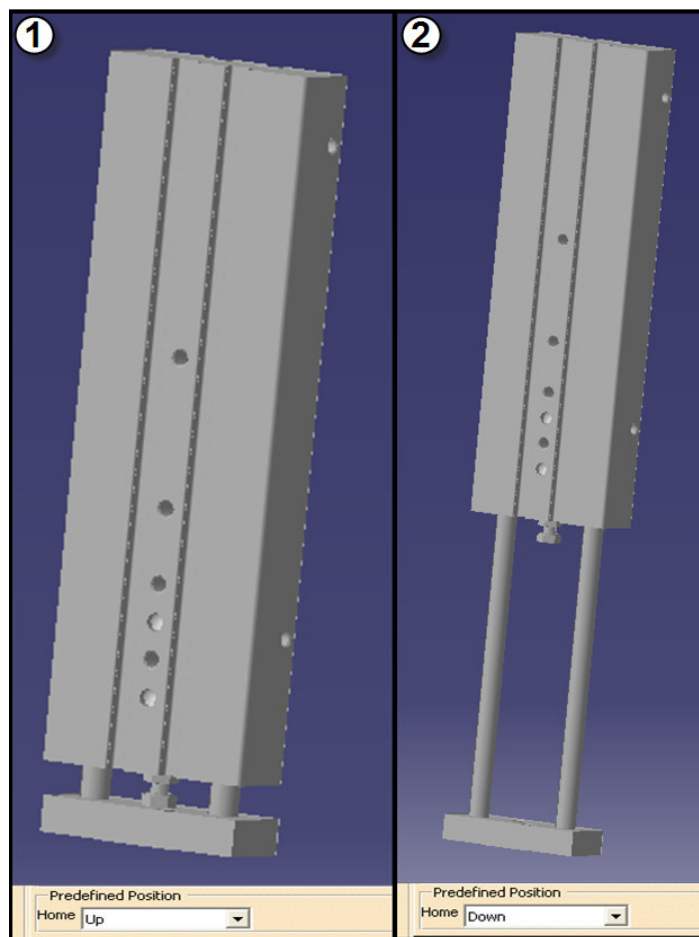


Figure 3.6 Model of a cylinder in two home positions

3.2.2 Creating Control Logic

In order for a device to imitate the behaviour of a physical counterpart or cause several devices to cooperate with each other within a digital factory, control elements must be allocated to it. Figure 3.7 shows that control elements or control logic can be divided into two types, namely device logic and control logic [62] [63], where control logic divides into internal and external control logic.

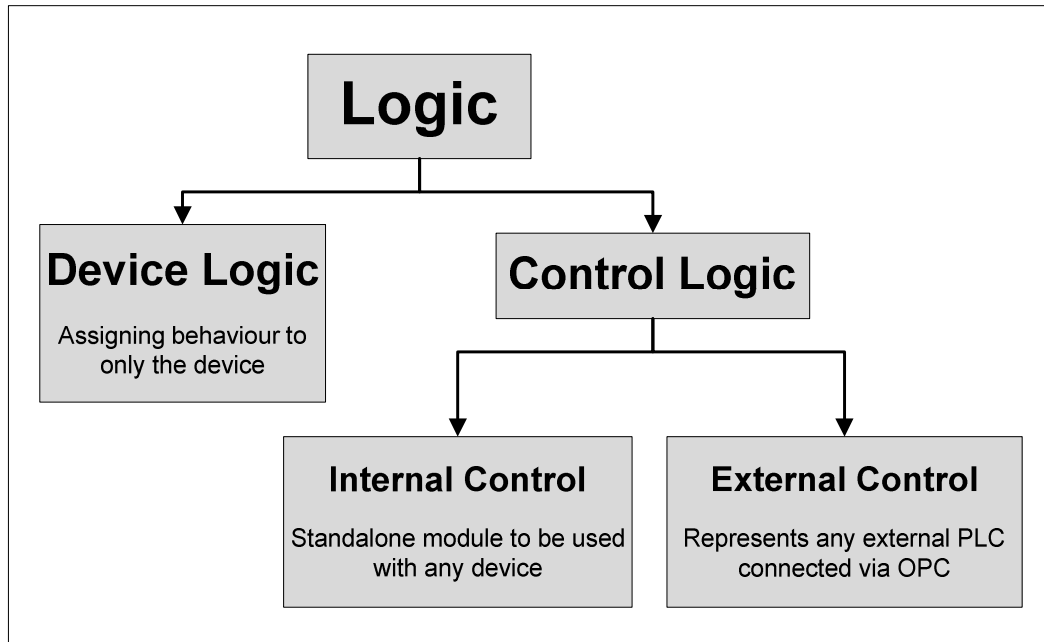


Figure 3.7 Description of control allocation

Firstly, device logic which is also known as internal logic (IL), assigns unique behaviour to the devices created in the previous section (make it smart devices). The internal logic monitors the input and output ports (IO ports), then perform tasks depending on the IO port conditions. Basically, IL assigns distinctive behaviour to each device and controls it via the IO ports. On the contrary, control logic is a standalone control block, which can be used to control various smart devices. To clarify, internal control can be seen as a virtual PLC (local virtual machine) connected to devices to emulate the behaviour of a real PLC. With external control on the other hand, the devices are connected to a real PLC via OPC (Ethernet). However, for the control elements to have intelligence and behaviour, a program to define it must be written and compiled. This is achieved by using functional block diagrams (FBD) or sequential function charts

(SFC), in conjunction with device tasks. The creation of both control and device logic are achieved in a similar manner and for the purpose of this study, only SFC programming will be examined. The steps to accomplish it follow below.

Referring to Figure 3.8 (at number 1), a control module must be created under the product tree. Afterwards, ports and signals are created, and the direction of these specified. An empty logic block is now generated. Next, the behaviour of the logic block is specified with the SFC editor [64]. From Figure 3.8 (at number 2) it can be seen that SFC uses squares to represent steps inside the behaviour. The steps are then connected by conditional transitions, symbolized by the lines between the steps. If a condition is true, the current step is terminated and the next step in the sequence will be active. In addition, optional “call action” tags are connected to each step. These actions include the calling of a task or changing the values of IO ports and signals. The SFC example in the figure shows the internal logic of a pneumatic cylinder. The input ports of the cylinder are monitored; then—depending on the transition condition—a corresponding task is called and the cylinder moves to the matching position. Onwards, control logic and internal logic can be assigned to all devices using the same method as described. Afterwards, the logic is built (compiled) and simulated to verify that the behaviour operates accurately. On completion, the various devices can be connected (mapped) to a control element, and therefore a digital factory is constructed.

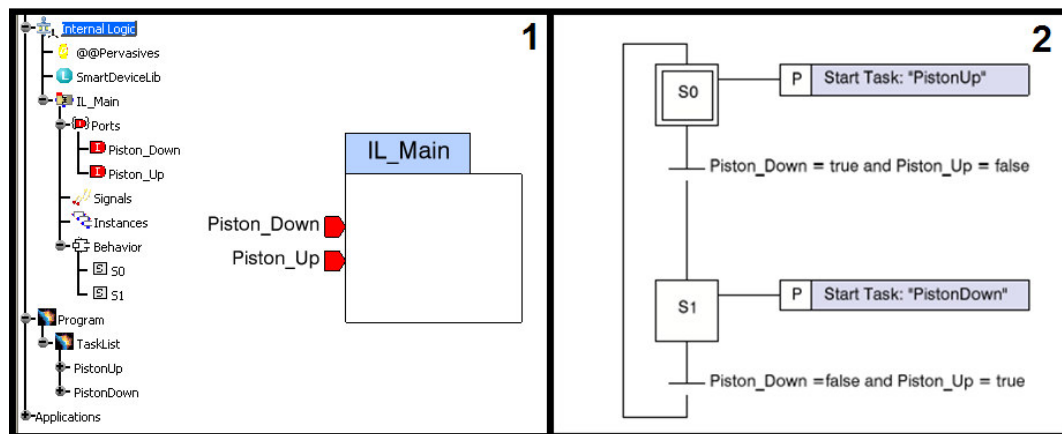


Figure 3.8 Control block with internal SFC logic

3.2.3 Process Plan and Simulation

In industry it is important to manufacturing companies to verify how their systems will operate well ahead of time. By commissioning a process simulation, manufacturers can visualize system tasks and through-put, undertake collision analysis, validate “what if” scenarios, and determine the effect of reconfiguration on a system. Therefore, planning a system is done more efficiently. Process planning within the DELMIA environment enables manufacturers to test a system without the use of control logic. A sequence of activities is defined in a process plan, enabling a process engineer to validate the behaviour of a work cell.

The following are the steps engaged to acquire a process plan:

Before a process plan can be obtained, the capabilities of a system and a sequence of activities must be provided. Activities consist of verified device tasks as well as other operations like move and delay activities. A process plan is used to plan and verify the sequential order in which subprocesses must be executed [65] [66] [67]. Further, activities can either be routed as series or concurrent tasks, validating the operation of a single task at a time or numerous tasks simultaneously. This is made possible by the use of PERT and Gantt charts. Referring to Figure 3.9, a PERT chart presents a graphical illustration of a process as a network diagram consisting of nodes (squares) representing subprocesses, linked by directional lines. The direction of the arrows on the lines indicates the sequence in which the tasks must be executed. The sequence of the subprocesses can be altered, simply by deleting directional lines, reordering the nodes and reconnecting the lines. The figure further shows how subprocesses can be defined by containing other subprocesses. However, the time taken to execute a process is defined by a Gantt chart. Referring to Figure 3.10, Gantt charts are a type of bar chart that shows the start, finish and the time duration of a process or subprocess. It can be altered to vary the process time by stretching the time bars for each subprocess. Various subprocesses are executed at different time durations—and by varying the process time, the speed of the tasks will be altered, synchronizing tasks to finish simultaneously. Lastly, by running a process simulation of the entire system, the

sequence in which tasks are executed can be observed which can further verify that the process plan is flawless.

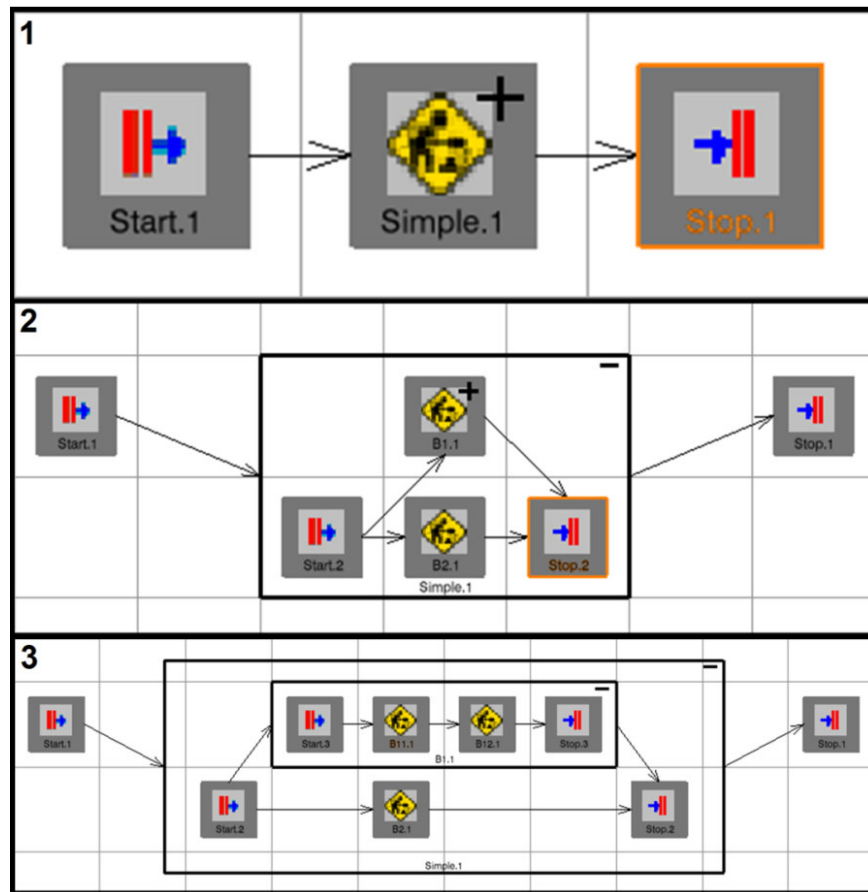


Figure 3.9 Example of a PERT chart

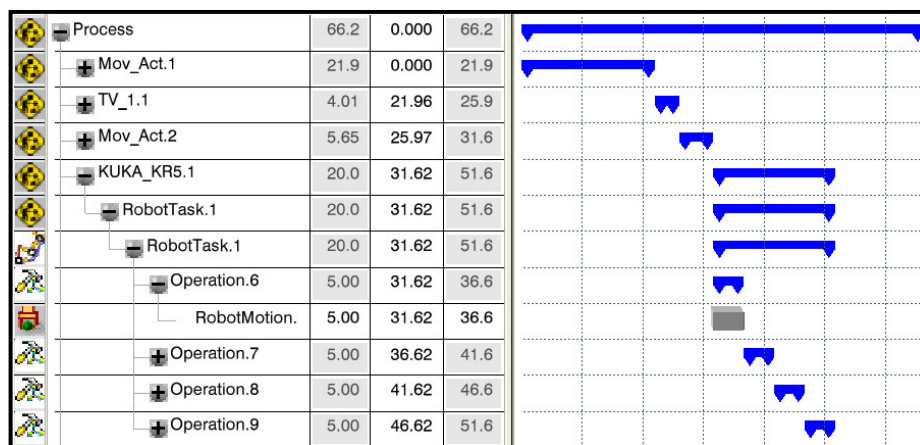


Figure 3.10 Example of a Gantt chart

3.2.4 Creating an Execution Environment and Virtual Commissioning

In contrast to a process plan, an execution environment uses control logic instead of process activities to validate the behaviour of a system [52]. Before an execution environment can be created, some work must be done ahead of time. This includes the installing of an OPC server and the set up of the support drivers to communicate with the physical devices used (explained in Chapter 4). Furthermore, all the smart devices to be used must be completed (section 3.2.1) and the control element (section 3.2.2) to run it must be programmed beforehand. Fortunately, DELMIA has a large selection of ready-built devices from a variety of vendors in their robot library that can be imported directly into the environment which expedites the process. Furthermore, the validation of a system using control logic is referred to as virtual commissioning. Virtual commissioning can be obtained using two methods. Firstly (Figure 3.11), the digital factory can be connected to a virtual PLC, to validate the performance of the system and the control logic inside the environment. Secondly, the digital factory can be connected to an external real PLC, validating the software used in the physical PLC and predicting how the physical system will react.

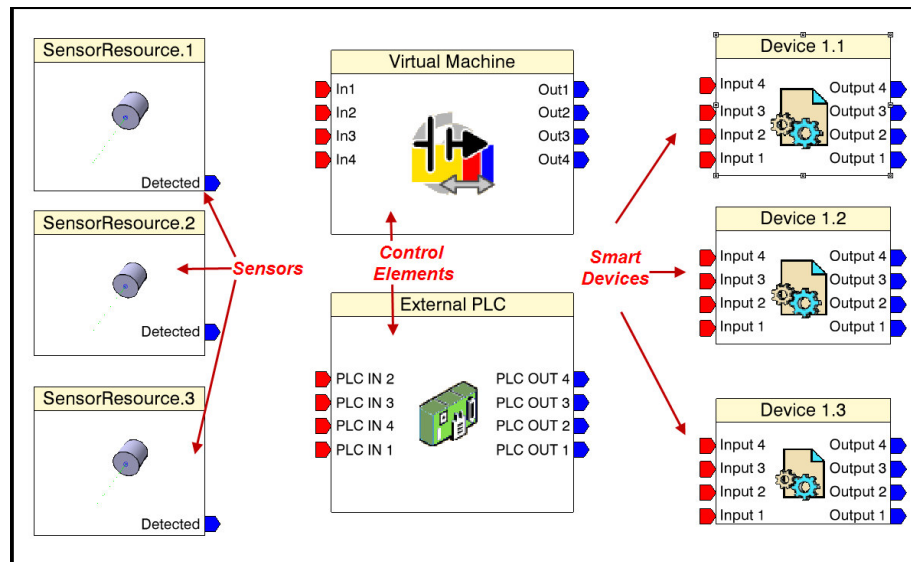


Figure 3.11 Device Control Connection Editor

An execution environment can be created, by using DELMIA’s “CSM Device Control Connection” workbench [68]. To describe this it is helpful to mention that, an execution environment is a virtual workspace where multiple elements are

gathered to function as one unit with a common purpose. These elements consist of control logic, device geometry and virtual sensory (Figure 3.11). The intention of this environment is to test how a virtual factory would behave with numerous variations of control logic and control types. Following below is an implementation example of the usage, and how to create an execution environment.

Firstly, the initial setup of the environment must be completed. All the smart devices (also known as resources), must be imported into the workspace and moved into place. Figure 3.12 shows how an environment is gradually being built and how further features are added to give a “real life” feel to the environment. After aligning all devices, it is necessary to mount all the EoA tools, by using attachments. An attachment is made between a resource and a tool in a parent-child approach, where the resource is the parent and the tool is the child (similarly to section 3.2.1.2). At this point the geometry part of the setup is complete and can be saved as a product.

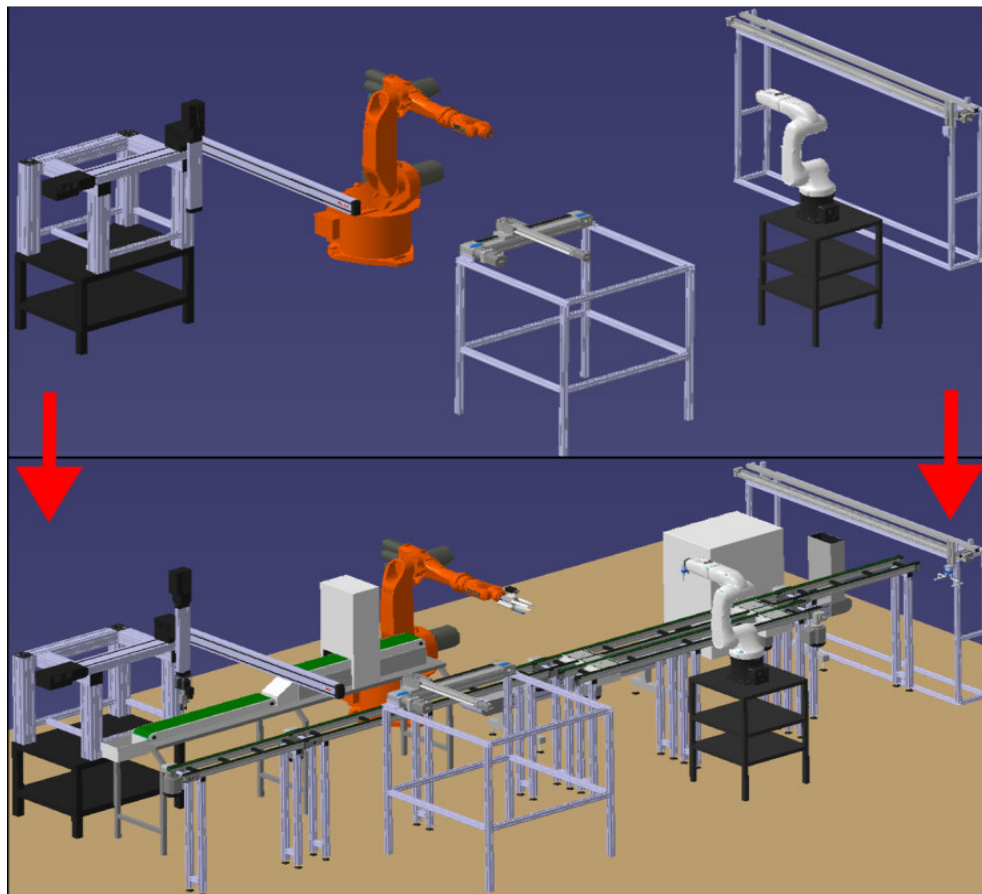


Figure 3.12 Virtual system built in DELMIA

Afterwards (see Figure 3.13), an execution environment is created under the product tree, whereafter devices and control elements can be entered into the environment. Given the fact that an assembly system can consist of multiple work cells, it is not necessary to include all the devices that are in the product tree into the execution environment, since only the included devices (a single work cell) will be part of the emulation.

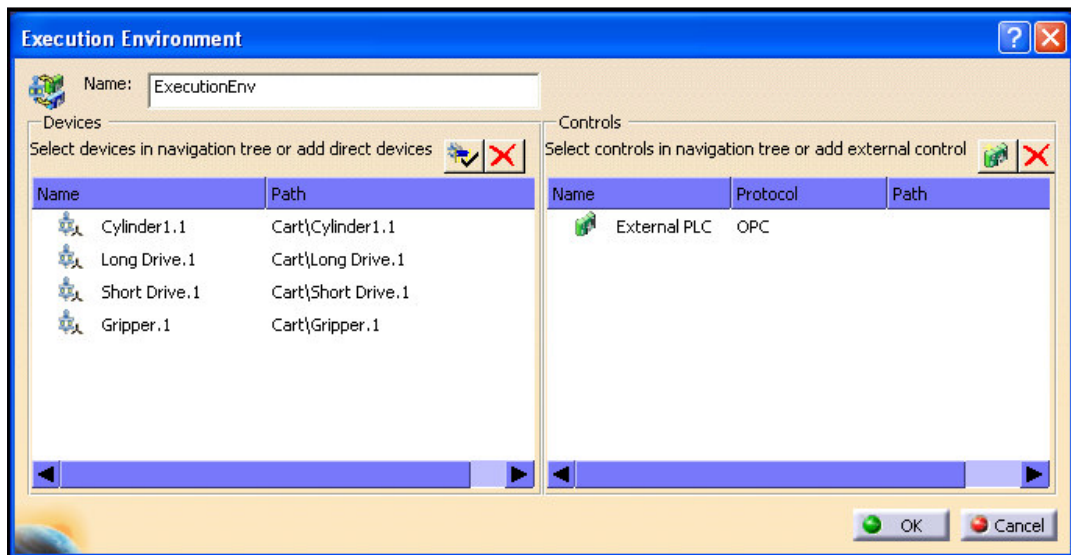


Figure 3.13 Execution environment setup window

Next, it is necessary to set up the communication to obtain an “external control connection” between the execution environment and the OPC server (OPC server is handled in Chapter 4). This is done by selecting OPC as the protocol type and then opening the “external PLC properties” window (Refer to Figure 3.14). Next, enter the path (address) and name of the server, and then connect to the server. If connected successfully, the user will be able to browse the server to obtain the address path for the PLC, as well as the OPC tags within it. The needed tags are selected, the type (Boolean for example) and direction (in, out or bidirectional) specified, and then the signal quality of the tags are checked. After this setup, a control block representing the external PLC with all the selected IO ports will appear in the “device connection editor”. At this stage the PLC is created virtually and can be used inside the environment.

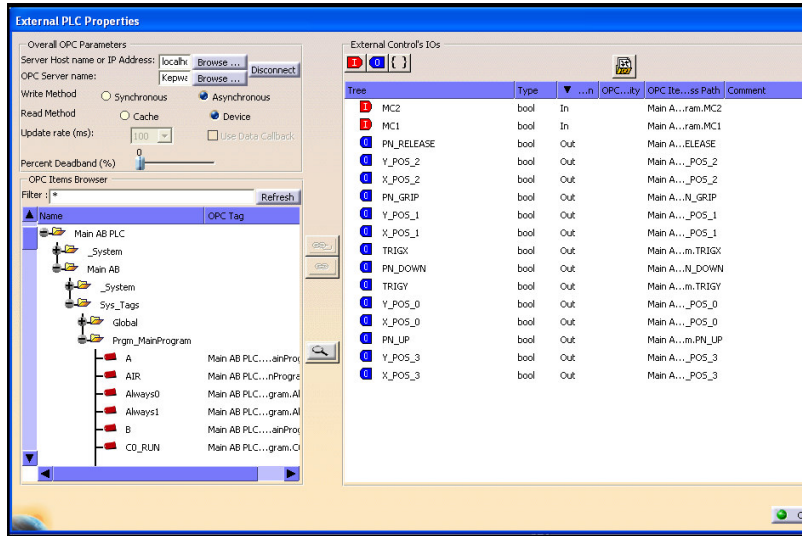


Figure 3.14 OPC setup for external PLC

Afterwards, the devices and control elements are interconnected by using the “device connection editor” workbench. Figure 3.15 shows how devices are represented by blocks with selectable IO ports and how they are connected to control elements by drawing lines (mapping) between the input and output ports (signals). Additionally, a human machine interface (HMI) can be included into the system to interface between the user and the PLC [69]. After all connections are made, the environment can be built (compiled) and checked for errors. After device and connection errors are rectified, the execution environment can be simulated and predictions can be made. In conclusion, the setup for virtual commissioning is obtained.

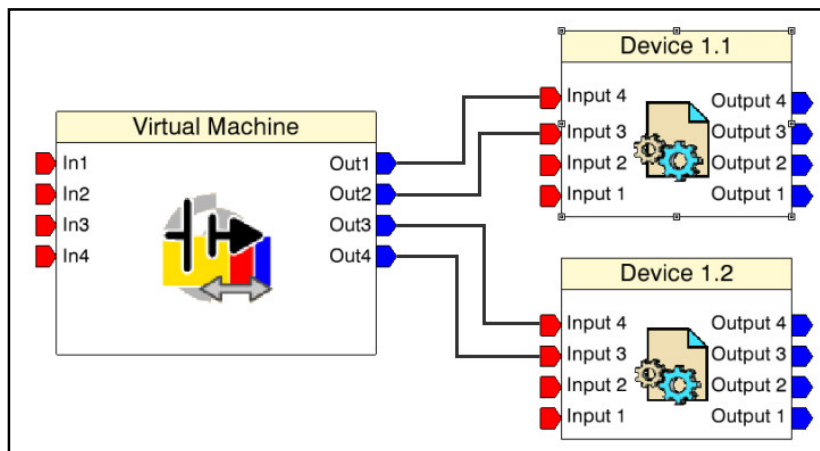


Figure 3.15 Interconnections using connection editor

3.3 Solution to Conveyor Limitation in DELMIA

3.3.1 Introduction

A limitation involved in DELMIA V5R20 being used in the project is that it does not support conveyors as a smart device. An additional solution is available in DELMIA Quest, but at additional cost to company. DELMIA has an option to create a conveyor run, but the feature is intended for space reservation. Furthermore, when conveyor geometry is downloaded from a vendor's website, it is normally in a rigid joint form. This causes a situation where no moving parts are present whatsoever. In short, the conveyor geometry cannot be used to emulate its physical counterpart.

To clarify the dilemma, if a pallet is loaded onto the conveyor from a pallet feeder, the pallet will remain stationary on the virtual conveyor; where on the contrary, a pallet will move along the real system conveyor in the same situation. Moreover, if a pallet reaches a stop-gate, the virtual environment will not detect that the pallet has reached an obstacle and will continue to move through the obstruction.

To remedy this predicament, a means to transform the conveyor into a smart device must be acquired. This will enable the conveyor to move pallets present on the conveyor and stop pallets when reaching a stop-gate, allowing the conveyor to imitate the operations of the real system conveyor. Thus, the following solution is developed.

The solution to the quandary can be divided into two sections. First, the geometry of the conveyor must be tailored to facilitate a moving mechanism to substitute the belt section of the conveyor. Secondly, the conveyor must be given behaviour by developing internal control logic. In conjunction, the latter replicates the operations of the real system conveyor and the result is shown in Appendix A.

3.3.2 Conveyor Geometry

Due to the fact that the conveyor geometry is obtained in rigid form, the following modifications must be made to the geometry to accommodate a moving functionality.

Firstly, by referring to Figure 3.16, the conveyor is fitted with stop-gates at predetermined positions where pallets must be stopped. The stop-gates are custom-built smart devices and need no modifications. Next, a replica of a system pallet (ghost pallet) is built and then assembled to the conveyor geometry to form a prismatic joint. Furthermore, the joint limits are specified as the length of the conveyor, with the result that the ghost pallet moves on the entire surface of the conveyor, with the result that the ghost pallet moves on the entire surface of the conveyor. In addition, the ghost pallet is made transparent and will appear invisible to the observer. This provides the moving mechanism, which is needed to imitate the moving action of the conveyor belt section.

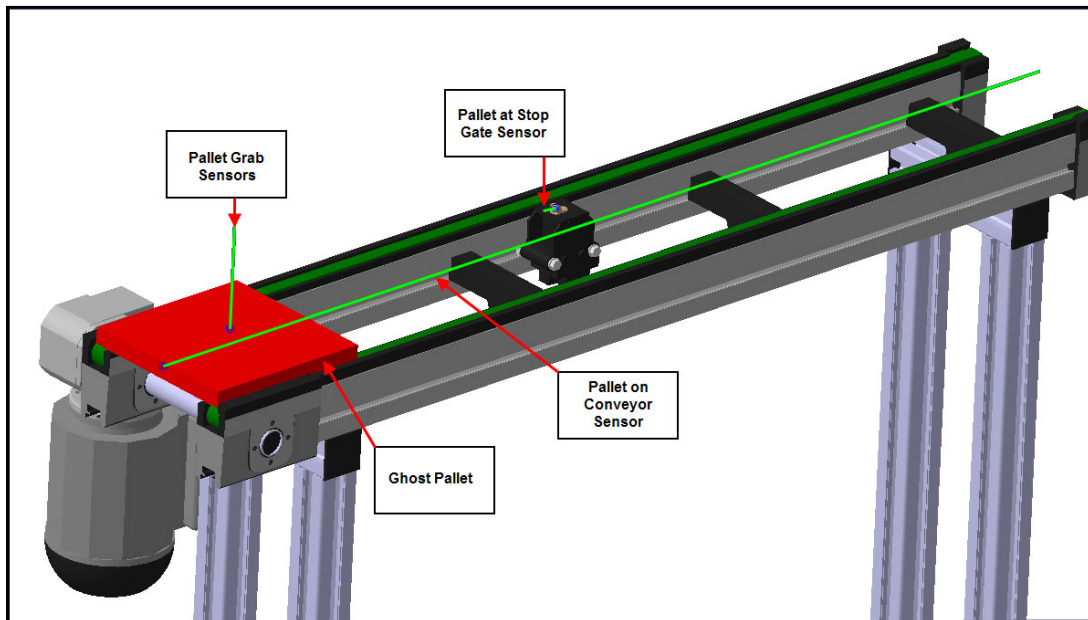


Figure 3.16 Conveyor with resource sensors

In addition to the moving mechanism, virtual resource sensors which are available in DELMIA must be placed at strategic locations along the conveyor. These sensors, as seen in Figure 3.16, informs the control logic of the conveyor about the presence and location of the pallets.

Referring to Figure 3.16, the first sensor is placed along the path the pallet must follow on the conveyor. Its purpose is to sense if a pallet is present on the belt section and actuate the movement if the conveyor motors are switched on. If both the presence sensor and the motor outputs are true, the ghost pallet will start moving as if the pallet is driven by the conveyor belt.

In addition, a sensor is attached to the ghost pallet. The purpose of this sensor is to sense and grab pallets, enabling the ghost pallet to drag along what is sensed, creating the illusion that the pallet moves.

Lastly, a sensor is attached to the stop-pin of the stop-gate. While this sensor is false, the pallet is free to move from the one side of the conveyor to the other. However, if the pallet travels within close proximity of the stop-gate, the sensor will detect that the pallet is touching the stop-gate and signal true. When the sensor is true, the movement of the ghost pallet is stopped, with the result that the pallet remains stationary as if the stop-gate is restraining it to move. On the contrary, if the stop-gate is activated, the stop-pin along with the sensor will retract, resulting in the situation that the sensor will not detect the pallet anymore. Afterwards, the ghost pallet will continue to move again, emulating that the obstruction is removed. The geometry of the conveyor is now tailored to imitate the operation (movements) of the real conveyor.

3.3.3 Conveyor Internal Logic

Recalling from section 3.2.2, SFC code starts at an initial step (S0) and executes the succeeding steps in an order determined by transitioning conditions and repeats. The rate at which these steps are scanned and executed is referred to as the scan rate or cycle time. By obtaining the cycle time, the movement of a mechanism can be manipulated to move at a certain speed [70]. This method is used to move the pallet on the conveyor and make it seem realistic.

To provide the conveyor mechanism with the necessary logic behaviour, the SFC code as shown by Figure 3.17 is implemented to operate the conveyor actions. Firstly, the routine starts and the initial step is activated. At this step, the speed constant at which the conveyor must move is obtained and the “motors on” condition is monitored. If “motors on” is false, the conveyor must remain at a standstill. On the contrary, if “motor on” is true, the next step is activated and the following variables are manipulated simultaneously. First, the cycle time is obtained from the simulation. It is then used in a formula along with the speed constant to obtain the step value at which the conveyor mechanism must move. The step value obtained is added to the current step value, and updated by writing it to the position value of the conveyor mechanism. This provides the

illusion that the conveyor moves at a constant speed. Subsequent to the variable manipulations, the position value of the conveyor and the resource sensors are monitored to determine which conveyor action to follow. To clarify, if the position value is less than the maximum conveyor length, a pallet is present on the conveyor and if the pallet is not touching a stop-gate, then the pallet is grabbed by the ghost pallet and will move until one of these conditions is not met. On the contrary, if the pallet moves across the conveyor, the position value will be greater than the maximum length of the conveyor with the result that the pallet is no longer present on the conveyor. Afterwards, the pallet is released by the ghost pallet; the mechanism is reset to its initial position and is then ready to move the next pallet.

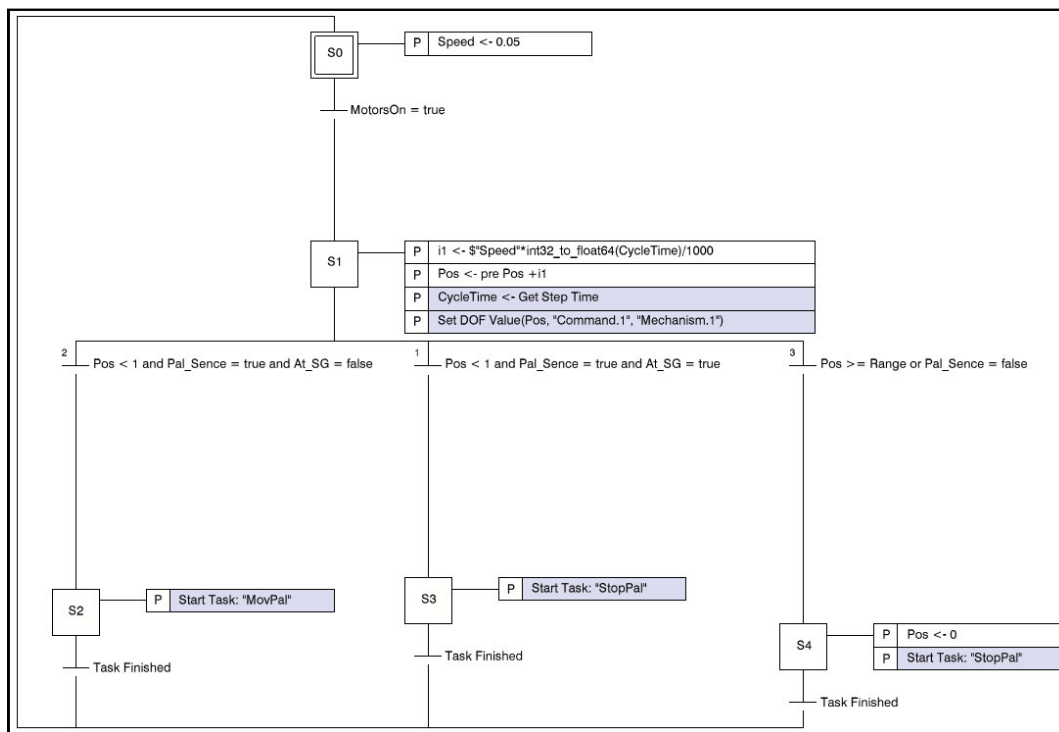


Figure 3.17 SFC code implemented in conveyors

3.3.4 Summary

To summarize, by transforming rigid conveyor geometry into a smart device enables virtual conveyors to imitate their physical counterparts. In addition, this method provides a tool to emulate an assembly system successfully and is used as part of the tests done in Chapter 5.

3.4 Conclusion

Fundamentally, this chapter reveals the path taken to verify the operation and control logic of a modelled system through obtaining virtual commissioning. In addition, it provided clarity on how the infrastructure of the DELMIA environment works; how parts, assemblies and mechanisms are built to form the geometry used in the system; and how to create smart devices by developing internal logic and adding it to this geometry. In addition to system geometry, various methods of verification were identified, which enabled the prediction of system behaviour and the validation of internal and control logic without initially building the actual system. One of these methods verified the operation of the system using a process plan, which enables a system designer to reserve space for machinery, visualise the operations of this machinery and perform crash analysis on the system without risking damage to the system components. The remaining two methods focused more on validating the control and internal logic of system devices and how the system reacts to the applied virtual or real PLC code. These methods are extensively used to debug initially developed control logic to validate a system as soon possible and analyse changes to be made to the control logic of a running system without pausing actual production, thus limiting downtime. To aid the latter, how to set up and connect to an OPC server, and mapping OPC tags and IO ports amongst devices in the environment were also discussed. This formed the basis for communication between the environment and the implemented control logic. In addition, a solution to conveyor limitations were provided, which further contributed to the foundation needed to perform virtual commissioning. This solution enabled the animation of conveyor operations based on the signals sent to and from the control logic. To conclude, it showed that by using DELMIA to establish virtual commissioning; proper initial planning can be performed; apparent design faults can be resolved early in design stages; analysis can be done to validate changes to a system; and it can be determined if it is actually profitable to build a system—thus, build it right the first time or not at all. The exposition above reveals that DELMIA provides a great tool for overall system verification.

Chapter 4 Methodology

4.1 Introduction

The research problem at hand insists that it would be beneficial to manufacturers if a RAS with enhanced methods of control is developed. This chapter reveals the methodologies undertaken to develop such a system. These methodologies include the following: identifying the system layout, the intended operation and product flow, obtaining the hardware and software components, and finding a means to intercommunicate between components. In addition, safety precautions must also be considered to maintain a secure setting. Everything considered, this chapter provides the methods to accomplish the development of the system.

4.2 System Hierarchy

Figure 4.1 shows the system hierarchy and an overview of the interconnections between the system components. Two types of communication protocols were investigated and utilized in the system, namely Ethernet and DeviceNet. Ethernet was used for communication between the (MAS), OPC server, an operator panel (HMI), a programming terminal (PC), and the system main controller. On the other hand, DeviceNet was used (via a remote DeviceNet module) to communicate between the system controller, the KUKA articulated robot and the gantry crane. In addition, the Cartesian assembly robot, proximity sensors, and pneumatic actuators are wired directly to the IOs of the main controller.

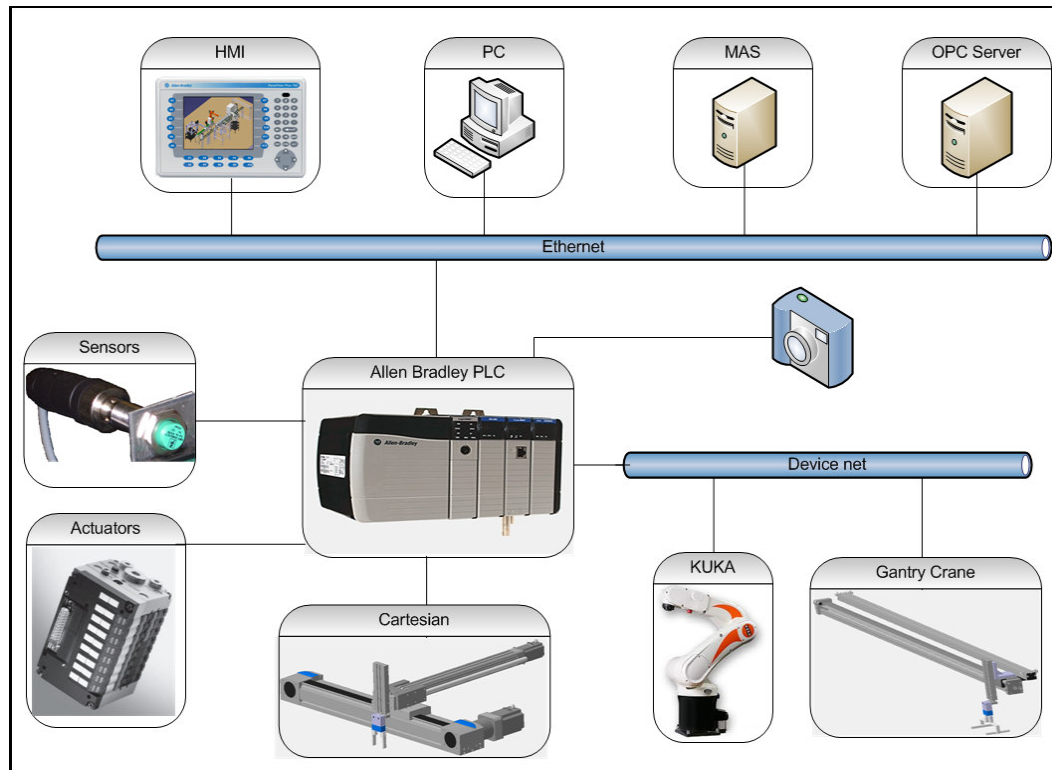


Figure 4.1 Network and component connection layout

4.3 Operation and Product Flow

Figure 4.2 shows the current layout and material flow, indicated by the red arrows, of the system at the RGEMS research laboratory. Initially, parts are fed into the system at node 1 and then pass through a visual inspection point at node 2. At this point a quality check is done to sort between good and faulty parts. Afterwards at node 3, the KUKA KR6 robot shown at node 4 picks up the parts that passed inspection and places them on the conveyor at node 5; the parts that failed inspection however, is left to be rejected. At node 5, parts again pass through a visual inspection point to be sorted by colour. At this point the Cartesian robot at node 6 picks up the sorted parts and places them into a parts feeder, which will supply parts to Cartesian robot at node 8. Meanwhile at node 7, pallets with trays mounted on top are fed into the system. These pallets proceed to the Cartesian robot at node 8, where matching features between products are built. Afterwards the pallets are conveyed to node 9, where again a visual inspection is done, whereafter the pallets are conveyed either to node 10 or 11. At this point in time the KUKA KR5-sixx at node 12 will build, for example, a

product A at node 10 and a product B at node 11. In contrast, if one of these conveyors should fail, the system must reroute the products to the working conveyor and build both on the same line. On completion, the products are conveyed to a gantry crane at node 13 where, finally, products are either rejected, reworked or distributed.

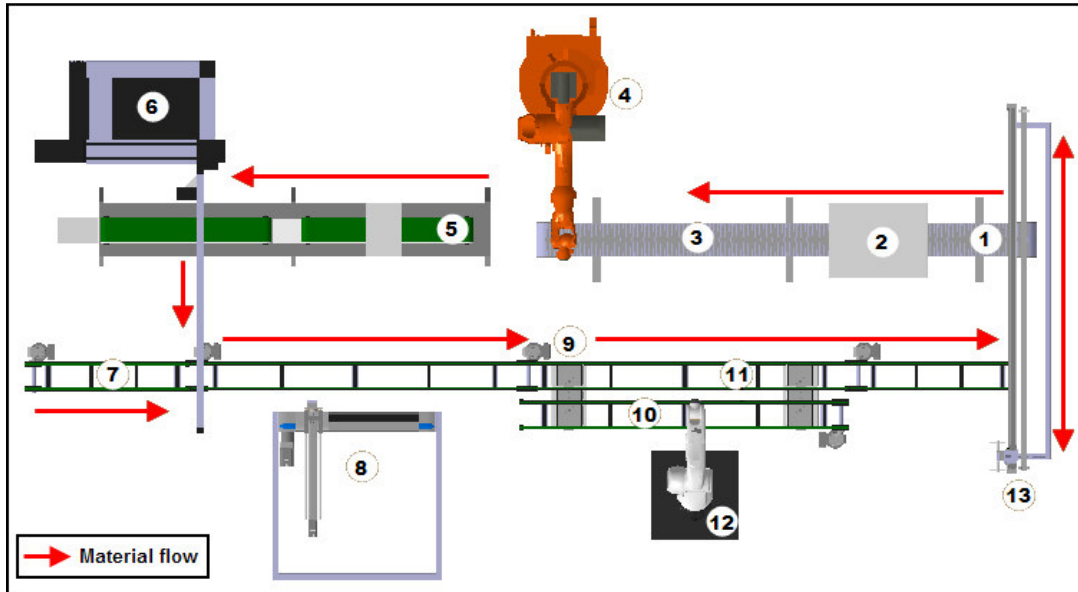


Figure 4.2 System layout and product flow

4.4 OPC and KEPserver

4.4.1 Background

The system utilized for this project in the RGEMS laboratory consists of multiple different devices from different vendors. Based on this diversity in system components, a communication platform to act as a standardized interface between the incompatible applications must be utilized. OPC is a feasible solution to be implemented in the system because of the fact that OLE for Process Control (OPC) is a proven, widely-used industry standard, compatible with multiple vendors' products. In addition to OPC being a widely used industry standard, OPC is the communication protocol used by the majority of PLM software packages to obtain virtual commissioning.

4.4.2 OPC

OPC was initially designed to provide a standard communication interface between Windows-based software applications and process control hardware (PLCs) [71]. OPC is a standards specification which resulted from collaboration between leading worldwide automation suppliers and Microsoft in 1996. It is based on Microsoft's OLE (Object Linking and Embedding) Component Object Model technology, which compels that data must be transferred in a standardized and usable format between applications. Furthermore, the standard defines reliable methods of accessing and manipulating field data from implemented factory floor devices.

Each application or client must implement one OPC compliant driver to access data from any OPC compliant server. In addition, these OPC servers provides the platform for software clients to access production data from process control devices, such as PLCs, and interact on a client-server base.

In conclusion, OPC forms the communication platform for software applications to communicate with factory floor hardware devices, including that all these devices are connected to the factory floor network (Ethernet).

4.4.3 KEPserver

Multiple devices have OPC servers of their own that are developed by the manufacturers themselves and are easily accessed by various OPC clients. Unfortunately, these OPC servers do not necessarily support the competition's product or has enough flexibility to add or modify the server to accommodate all vendors. However, KEPware offers a solution, namely KEPserver [72], which supports multiple vendors and have support drivers for numerous hardware devices. Additionally, if a device is not supported, additional plug-in drivers can be imported into KEPserver to accommodate these devices. Following below are the steps to set up KEPserver.

4.4.4 KEPserver Setup

Firstly, KEPserver must be installed on a server (computer), which is networked with all the devices used in the system. In addition, software drivers to support

the devices which are not included in the initial KEPserver package must be imported to establish communication with these devices.

On completion of the installation, the KEPserver application can be executed to start the server runtime. At this point, either a new server can be configured or an existing server (runtime) can be modified by adding or removing devices and tags [73]. In addition, channels and devices can then be created, by specifying the devices by vendor and model, as well as specifying the network paths and IP addresses. Referring to Figure 4.3, the main AB PLC is visible through the OPC driver and can be browsed to see the OPC tags inside the device. Moreover, these tags are either created manually or can be imported into the server if the PLC supports this feature.

Tag Name	Address	Data Type	Scan Rate
A	PROGRAM:MAINPROGRAM.A	Boolean	100
AIR	PROGRAM:MAINPROGRAM.AIR	Boolean	100
Always0	PROGRAM:MAINPROGRAM.ALWAYS0	Boolean	100
Always1	PROGRAM:MAINPROGRAM.ALWAYS1	Boolean	100
B	PROGRAM:MAINPROGRAM.B	Boolean	100
CO_RUN	PROGRAM:MAINPROGRAM.CO_RUN	Boolean	100
C1_RUN	PROGRAM:MAINPROGRAM.C1_RUN	Boolean	100
C2_RUN	PROGRAM:MAINPROGRAM.C2_RUN	Boolean	100
C3_RUN	PROGRAM:MAINPROGRAM.C3_RUN	Boolean	100
CART_BUSY	PROGRAM:MAINPROGRAM.CART_BUSY	Boolean	100
CART_DONE	PROGRAM:MAINPROGRAM.CART_DONE	Boolean	100
CART_FULL	PROGRAM:MAINPROGRAM.CART_FULL	Boolean	100
CART_LOADED	PROGRAM:MAINPROGRAM.CART_LOADED	Boolean	100
CART_READY	PROGRAM:MAINPROGRAM.CART_READY	Boolean	100
CHECK_SUCCEED	PROGRAM:MAINPROGRAM.CHECK_SUCCEED	Boolean	100
COMP_DONE	PROGRAM:MAINPROGRAM.COMP_DONE	Boolean	100
CON_RED1	PROGRAM:MAINPROGRAM.CON_RED1	Boolean	100
CON_RED2	PROGRAM:MAINPROGRAM.CON_RED2	Boolean	100
COUNT	PROGRAM:MAINPROGRAM.COUNT	Char	100
GANTRY_TRIG	PROGRAM:MAINPROGRAM.GANTRY_TRIG	Boolean	100
KUKA_DONE	PROGRAM:MAINPROGRAM.KUKA_DONE	Boolean	100
KUKA_FULL	PROGRAM:MAINPROGRAM.KUKA_FULL	Boolean	100
KUKA_LOADED	PROGRAM:MAINPROGRAM.KUKA_LOADED	Boolean	100
KUKA_PROG	PROGRAM:MAINPROGRAM.KUKA_PROG	Boolean	100
KUKA_TRIG	PROGRAM:MAINPROGRAM.KUKA_TRIG	Boolean	100
M0	PROGRAM:MAINPROGRAM.M0	Long	100
M1	PROGRAM:MAINPROGRAM.M1	Long	100
MC1	PROGRAM:MAINPROGRAM.MC1	Boolean	100
MC2	PROGRAM:MAINPROGRAM.MC2	Boolean	100
MOTOR_0	PROGRAM:MAINPROGRAM.MOTOR_0	Boolean	100
MOTOR_1	PROGRAM:MAINPROGRAM.MOTOR_1	Boolean	100
MOTOR_2	PROGRAM:MAINPROGRAM.MOTOR_2	Boolean	100
MOTOR_3	PROGRAM:MAINPROGRAM.MOTOR_3	Boolean	100
MOTOR_4	PROGRAM:MAINPROGRAM.MOTOR_4	Boolean	100
MOTOR_5	PROGRAM:MAINPROGRAM.MOTOR_5	Boolean	100
MOTORS_RUN	PROGRAM:MAINPROGRAM.MOTORS_RUN	Boolean	100
NOT_CLRD	PROGRAM:MAINPROGRAM.NOT_CLRD	Boolean	100
PAL_COUNT	PROGRAM:MAINPROGRAM.PAL_COUNT	Char	100

Figure 4.3 OPC tags in KEPserver

If tags must be created manually, the respective device under which the tags must be created is highlighted, and then tags are created individually or as groups. Furthermore (referring to Figure 4.4), tag properties can be modified and tags can be assigned user-definable descriptive names, as well as the tag

address located in the respective device. In addition, the data type, clients' read or write privileges and scan rate for each tag are specified.

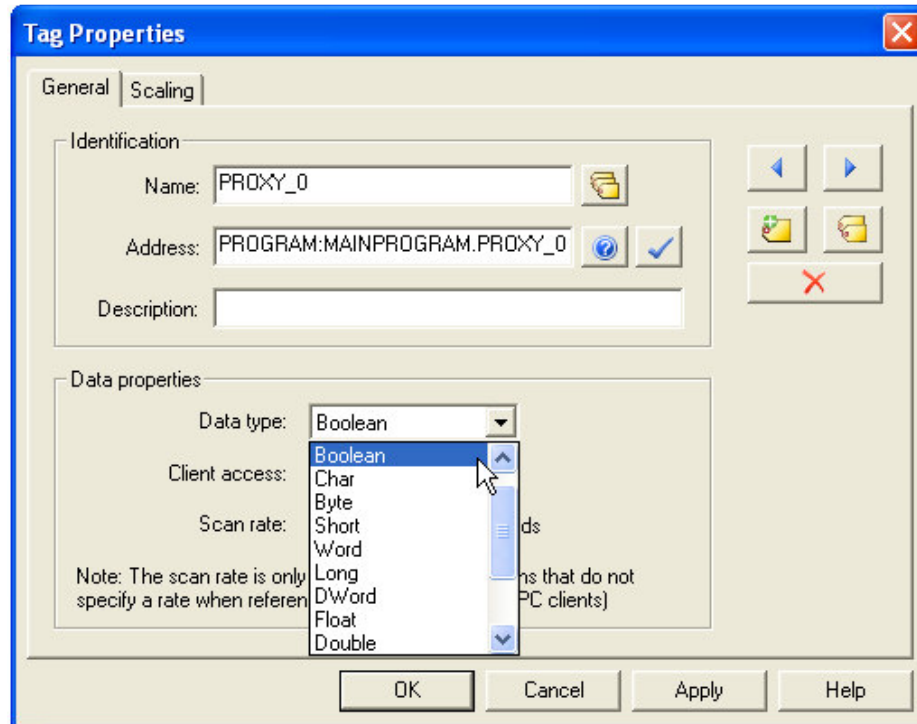


Figure 4.4 Modifying tag properties

Afterwards, the quality of the OPC tags is then confirmed by running the OPC quick client. This ensures that a good connection between the server and the various devices are established and that the data are able to update. In addition to quality checking, the quick client is also used for monitoring and manipulating tag data, to ensure the accurate operation of the implemented production system.

In conclusion, by utilizing KEPserver, multiple devices from different vendors can be interfaced and tested, without the need for expensive interfacing equipment, and, additionally, these devices are able to intercommunicate by using a standard data format.

4.5 System Components

This section discusses the components that are chosen to be utilized in the system, as well as why they were chosen. In addition to component choice, how these components are configured and controlled, along with their function in the system are also described. Moreover, the control software developed on device

level to operate these components is also an area under discussion. In short, this section shows how the selected components are implemented in the system.

4.5.1 Material Transport System

The choice in the TS1 [74] conveyor system from Rexroth is based on a study done within the RGEMS research group [75]. The study shows that the choice in this conveyor is based on cost, ease of assembling and rearranging sections of the conveyor, size, maximum payload (accumulative weight it can handle), and ergonomics.

The conveyor is built up from modular belt sections, which makes it possible to expand, change or rearrange the path of the conveyor. In addition to the modularity of the conveyor, the use of work piece pallets contributes immensely to the reconfigurability of the conveyor, in the sense that any product in a range can be built on the pallet. Furthermore (refer to Figure 4.5), each belt section has two green toothed belts (shown at A), which are guided by aluminium profiling. In addition to the guiding profile, these belt sections are powered by three-phase induction motors, which are connected to the PLC via contactors. Finally, the belt sections are placed on top of foot pieces (legs), to form the frame and structure of the conveyor.

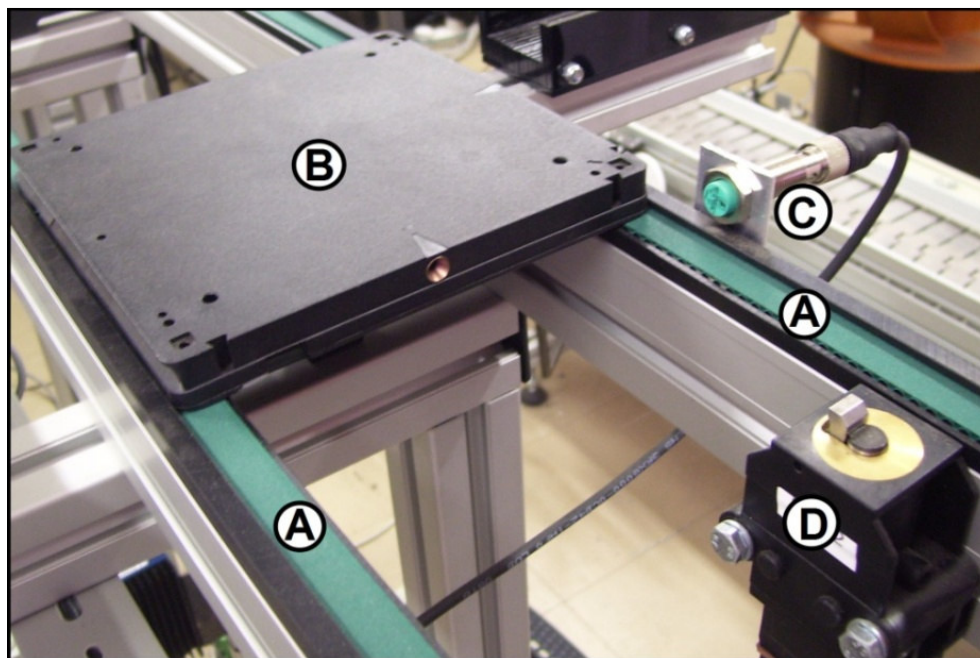


Figure 4.5 Conveyor system components

For operation refer to Figure 4.5. Shown at B, the work piece pallet is placed on top of the conveyor, where the pallet starts moving purely on friction from the belts. The pallet can then be stopped by pneumatic actuators, also known as stop-gates (shown at D). The default position for a stop-gate is “up”, meaning that it stops the pallets from continuing when it reaches the stop-gate. When a pallet is stopped by a stop-gate, the two belts beneath it continue to move, rubbing gently against the smooth and anti-static bottom of the pallet. To let the pallet pass, the stop-gate is pulsed for a second by the PLC, allowing the pallet to move past the stop-gate. In conjunction with stop-gates, inductive (sense metal) proximity sensors (shown at C) are mounted along the conveyor to sense if pallets are present at defined positions. The pallets are fitted with metal inserts situated on the front and side, which allows inductive proximity sensors to detect if pallets are in place. Using inductive proximity sensors in combination with metallic inserts, the system can distinguish between individual pallets present at a precise defined location. These sensors ignore the sides (non metal) of a pallet until it reaches the metal insert. This is particularly useful when pallets are stacked behind one another on the conveyor. In this situation where there is no gap/space between the pallets, the possibility exists that a different type of proximity sensor would always erroneously detect multiple pallets as one single pallet.

The length of the conveyor can be changed by adding or removing the modular belt sections. In addition to changing the length, it is possible to transfer the work piece pallet off the main conveyor onto a shunt conveyor system using a branching conveyor. Therefore, two products can be built simultaneously on the two parallel sections. This is illustrated in Figure 4.6, which shows two parallel conveyors, lane A (shown at A) and lane B (shown at B), where they are joined together by a tandem lift transverse conveyor (shown at C). The tandem lift transverse conveyor units are operated by pneumatics and have three operating positions namely idle, up and down. Firstly, when the transverse conveyor is in its default “idle” state, it remains stationary in the middle. Here it acts as a stop-gate and stops the pallets from continuing. Next, in the case the transverse conveyor is actuated in the “down” position, it moves downwards and allows the pallet to pass and continues straight on lane A. On the contrary, when the transverse

conveyor is in the “up” position, the platform on both lanes lifts simultaneously, acts as a bridge and transfers the pallet from lane A to lane B by the belt section C. The same procedure is followed when a pallet is transferred back to the main conveyor (lane B to A).

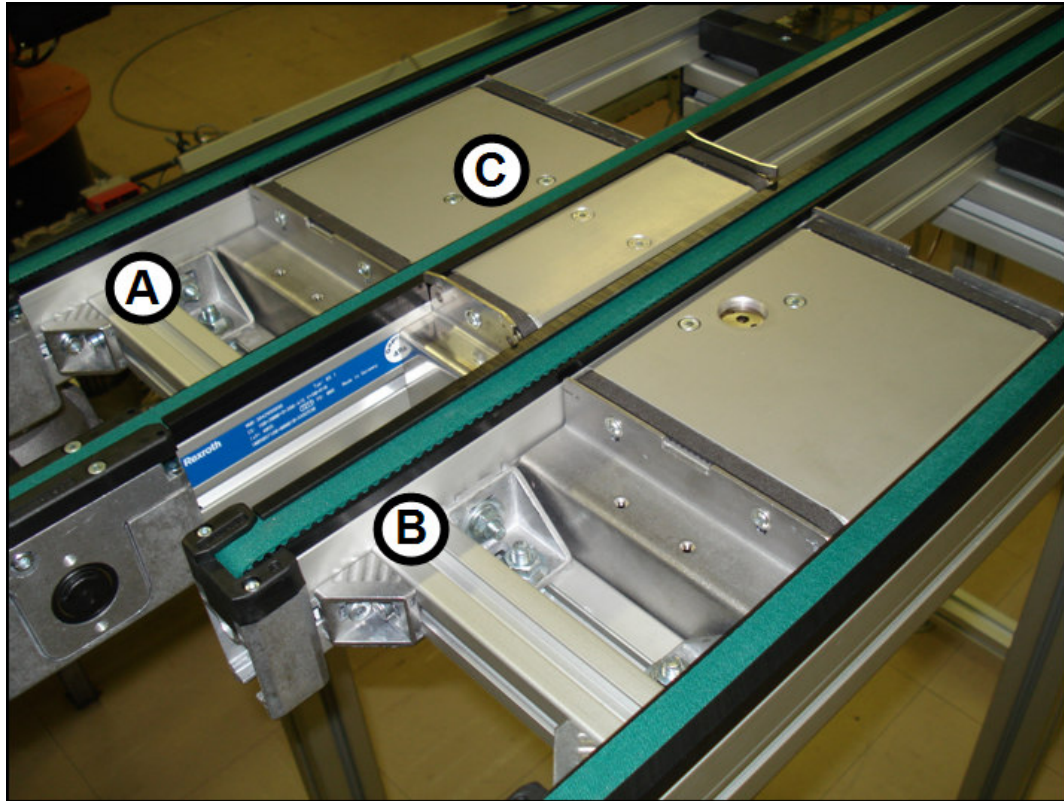


Figure 4.6 Tandem lift transverse conveyor

4.5.2 Pneumatic Parallel Gripper

Figure 4.7 shows the pneumatic parallel gripper which is used as an end effector in all the subsequently explained assembly devices. It uses pressurized air to operate and is used to grasp parts which are being handled by the assembly devices. It is chosen because it is readily available and it is a standard gripping tool.



Figure 4.7 FESTO gripper [76]

4.5.3 FESTO Cartesian Robot

Firstly, the function of the Cartesian robot in the system is to build all the similar features or patterns, which exist between different products and leave the rest of the placing tray empty. Secondly, the Cartesian is chosen because of the ease of operation; ease of configuration; easy integration with a PLC; and the extensive use of Cartesian robots in assembly operations.

4.5.3.1 Description

Referring to Figure 4.8, the Cartesian robot is a custom-built 3-axis (XYZ) assembly robot, which is built entirely of components manufactured by FESTO. It consists of two linear drives, which is mounted perpendicularly in respect to each other, to form the X-axis and Y-axis. Additionally, the linear drives are fitted with stepper motors, which are connected to and controlled by FESTO motor controllers [77] [78]. In addition, a pneumatic cylinder in conjunction with a parallel gripper are mounted perpendicularly to the Y-axis to form the Z-axis. Furthermore, the gripper is mounted with fingers, which are specially designed using CATIA and manufactured by the Centre for Rapid Prototyping and

Manufacturing (CRPM) at the CUT. This enables the gripper to pick the blocks from a part magazine and place the parts into the tray mounted on a pallet. Lastly, a frame is built using aluminium profile to mount and stabilize the linear drives. It is also used to enclose the moving axis of the robot and prevent injuries caused by impact.

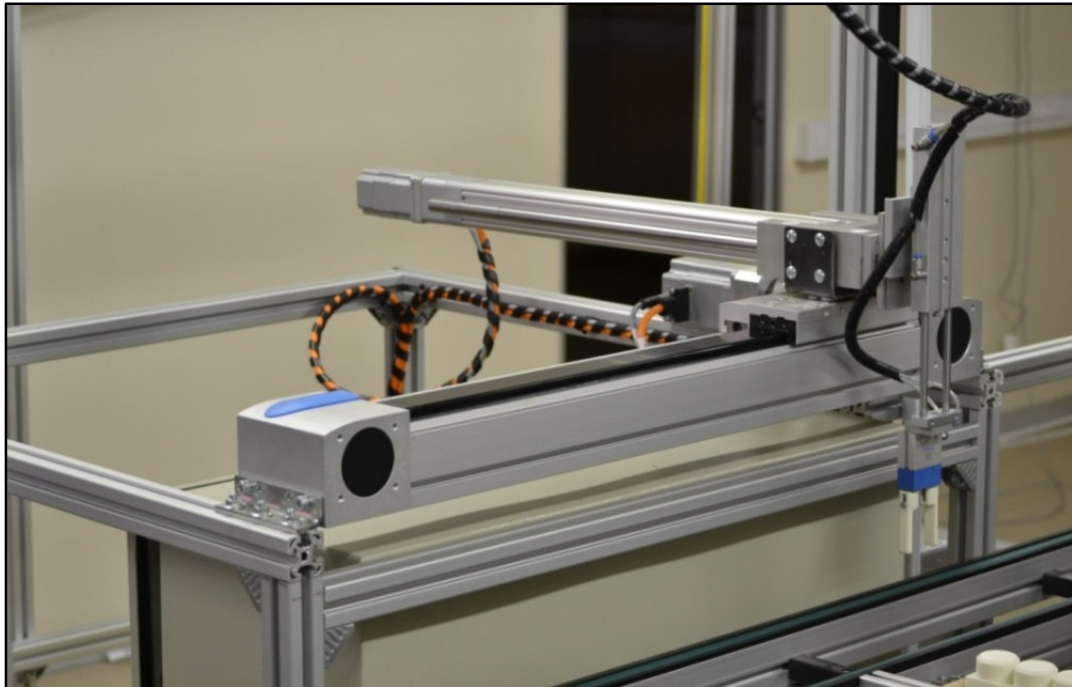


Figure 4.8 FESTO Cartesian robot

4.5.3.2 Setup and Calibration

Following below are the procedures taken to configure FESTO motor controllers:

After the installation of both drives and motor controllers, each axis of the Cartesian needs to be configured. Prior to configuring the motor controllers, the FESTO Configuration Tool (FCT) software must be installed on a computer intended for programming.

First, the motor controller is connected to the computer via a serial cable. Then the FCT software is opened and a software connection with the device established and a new project started. In contrast to starting a new project, an existing project can be opened or a project can be uploaded (read) from the corresponding device.

Next, Figure 4.9 shows the project start page, where the user should select the operating voltage; the type and size of the motor used; if a gearbox is present and the ratio of it, specifics of the type and stroke (working length) of the linear drive used, and finally the type of limit switches used.

In addition to choosing the components, the method of controlling the drive must be assigned. A selection between digital IO and analogue voltage control is available. Digital IO was selected as control method and is explained in section 4.5.3.3. However, analogue control use applied positive or negative voltages to control the direction of drive movement, where varying the magnitude of these voltages control the speed of the drive. Furthermore, the motor controllers have various different control modes, which include “single position set”, “link of position sets”, “synchronization”, and “jogging and teaching” [79]. Due to the simplicity of commissioning and operation, “single position set” was a satisfactory method of control which met the requirements of the system.

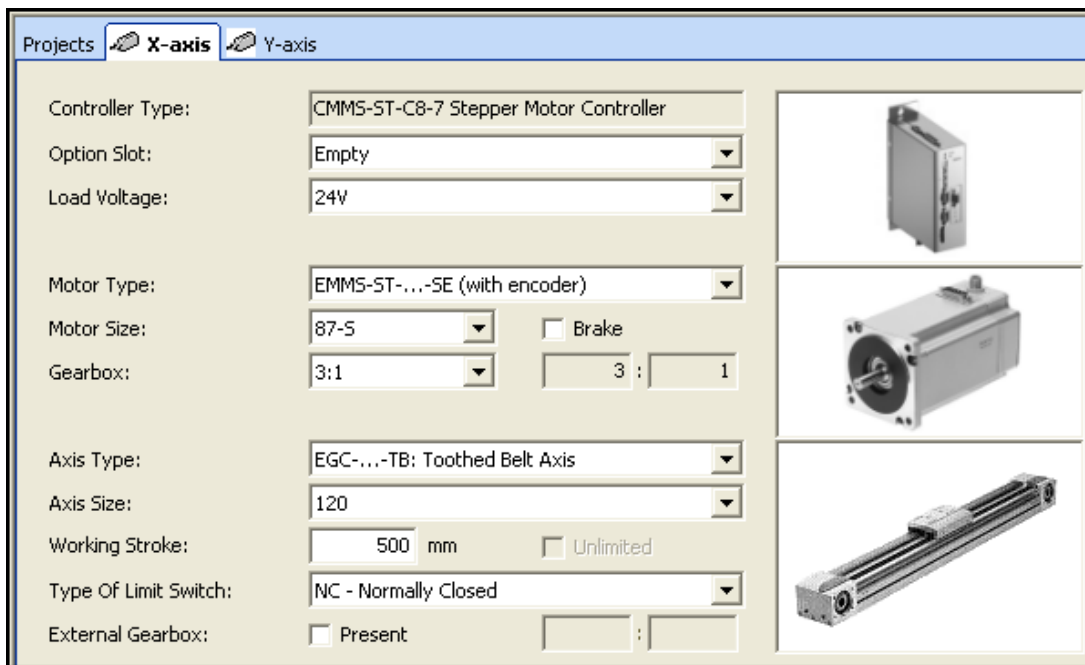


Figure 4.9 FESTO configuration tool software

In addition to control methods, the positions and motion attributes must be entered or taught. Referring to Figure 4.10 part (1), positions can be entered into the position set table by modifying the value in the “position” column, in the respective position (position number) row. Alternatively, the position row can be

selected, and by using the FCT software, jog the drive into position, then teach the position to the controller. Along with the physical position, the motion profile and command to be used can be defined. Moreover, Figure 4.10 part (2) shows the position profile table, which is used to assign velocity, acceleration and deceleration values to each profile, and optimizes the movements for best operation. Finally, the configuration must be downloaded (written) to the controller and stored to ensure correct operation after power up.

This should be repeated for every axis in the Cartesian robot, which is to be configured or recalibrated.

FCT	No.	Mode	Position [mm]	Profile	Command
	1	A	0.00	0	END
	2	A	-100.00	0	END
	3	A	-200.00	0	END
	4	A	-250.00	0	END
	5	A	-300.00	0	END
	6	A	-350.00	0	END
	7	A			

No.	Vel. [mm/s]	Accel. [m/s ²]	Decel. [m/s ²]	Smooth [%]
0	46.00	1.000	1.000	0
1	46.00	1.000	1.000	0
2	46.00	1.000	1.000	0
3	46.00	1.000	1.000	0
4	46.00	1.000	1.000	0
5	46.00	1.000	1.000	0
6	46.00	1.000	1.000	0
7	46.00	1.000	1.000	0

Figure 4.10 Position set table (1), position profile (2)

4.5.3.3 Operation

Figure 4.11 illustrates the operation of the FESTO motor controllers. When power is applied to the motor controller, a compulsory start-up sequence is needed from the PLC to put the controller in a “controller enabled” state, which entails setting the “Enable Power” bit, delaying for 200ms, followed by setting the “Enable Control” (enabling the control logic) bit. Afterwards, a homing run must be completed, for the controller to find the physical limits of the linear drive. The controller can be configured to do this automatically after “controller enabled”, as well as the direction (positive or negative), or waits to be instructed by the PLC. When homing is completed, the controller is ready to be operated.

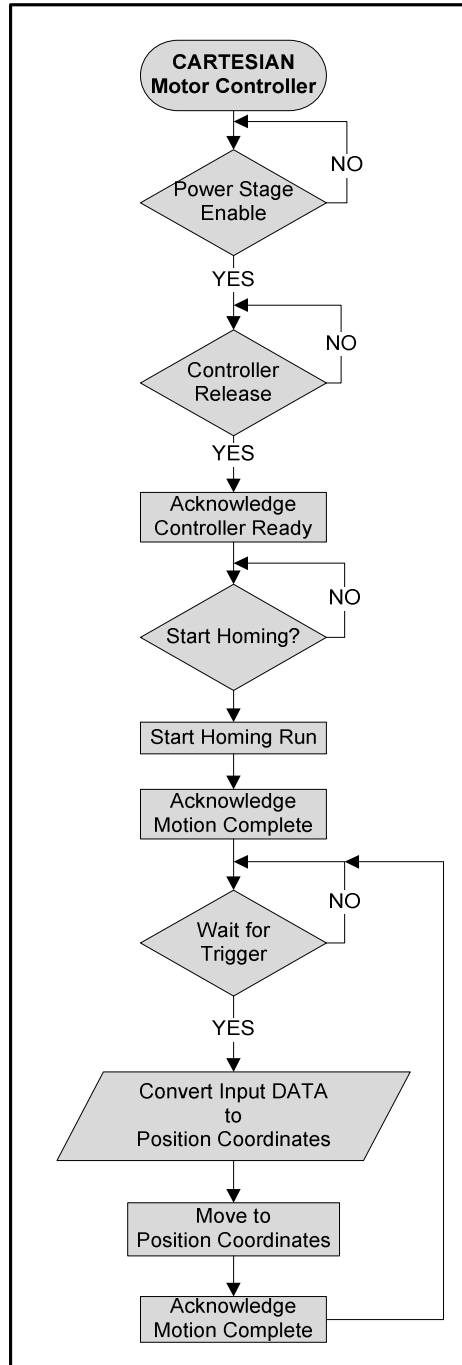


Figure 4.11 Flow of operation for motor controller

For operation, the controller uses (see Figure 4.12) four “position or record select” inputs, a trigger, and three user-definable outputs for acknowledgement, which is commonly defined as “Motion Complete”, “Acknowledge Start” and “Error”. Moreover, BCD (Binary Coded Decimal) code are applied to the position select inputs, which are used to retrieve coordinates from a predefined position set table

(see Figure 4.10) when the trigger is pulsed. The drive will start moving to the position and acknowledge with “motion complete” when the drive is stationary. Afterwards, the BCD value can be changed to select a different position from the position set table. As a result, the preceding procedure is repeated continuously to operate the linear drive.

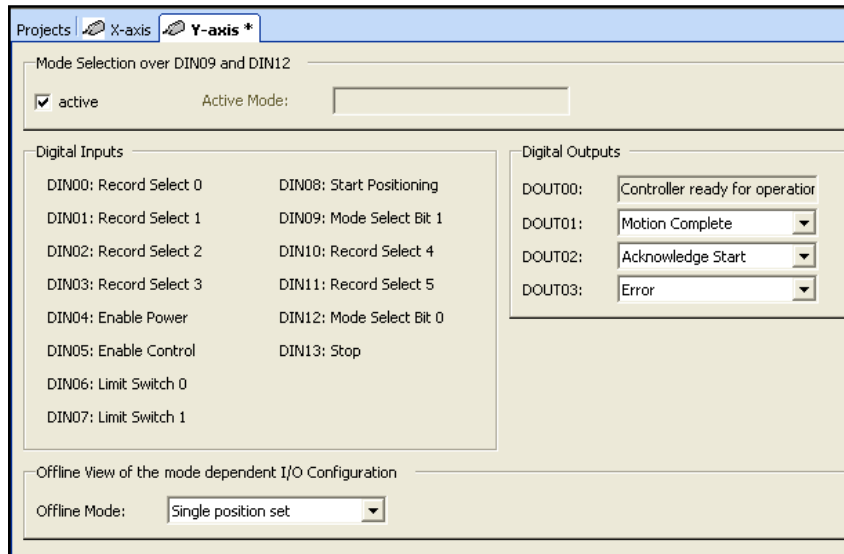


Figure 4.12 FESTO motor controller IO layout

4.5.3.4 Cartesian Controller Algorithm

The Cartesian controller algorithm is a modular software program (agent) running on a PLC, which is designed to monitor and control the operations of the motor controllers used to build the Cartesian. It can operate as part of the main PLC as a subprogram (thread) or as a standalone device running on an external PLC, interfaced with the main PLC. For the purpose of this project, the Cartesian controller algorithm is a thread inside the main PLC. Moreover, the Cartesian control algorithm receives production information (recipe) from the main process controller (main PLC) internally and handles the signalling to operate the motor controllers and actuators (gripper and cylinder).

At power up, the main process controller runs all the necessary prerequisite routines, and upon completion, enable the execution of the Cartesian controller shown in Figure 4.13. After the Cartesian controller starts to execute, it instructs the motor controllers to initialize by signalling the compulsory required sequence to enable the motor controllers and completing a homing run.

After initializing the motor controllers, the Cartesian controller acknowledges back to the process controller that the Cartesian is ready to operate, and waits for further instructions. When a pallet is in place, the process controller sends product data to the Cartesian controller and instructs it to start building.

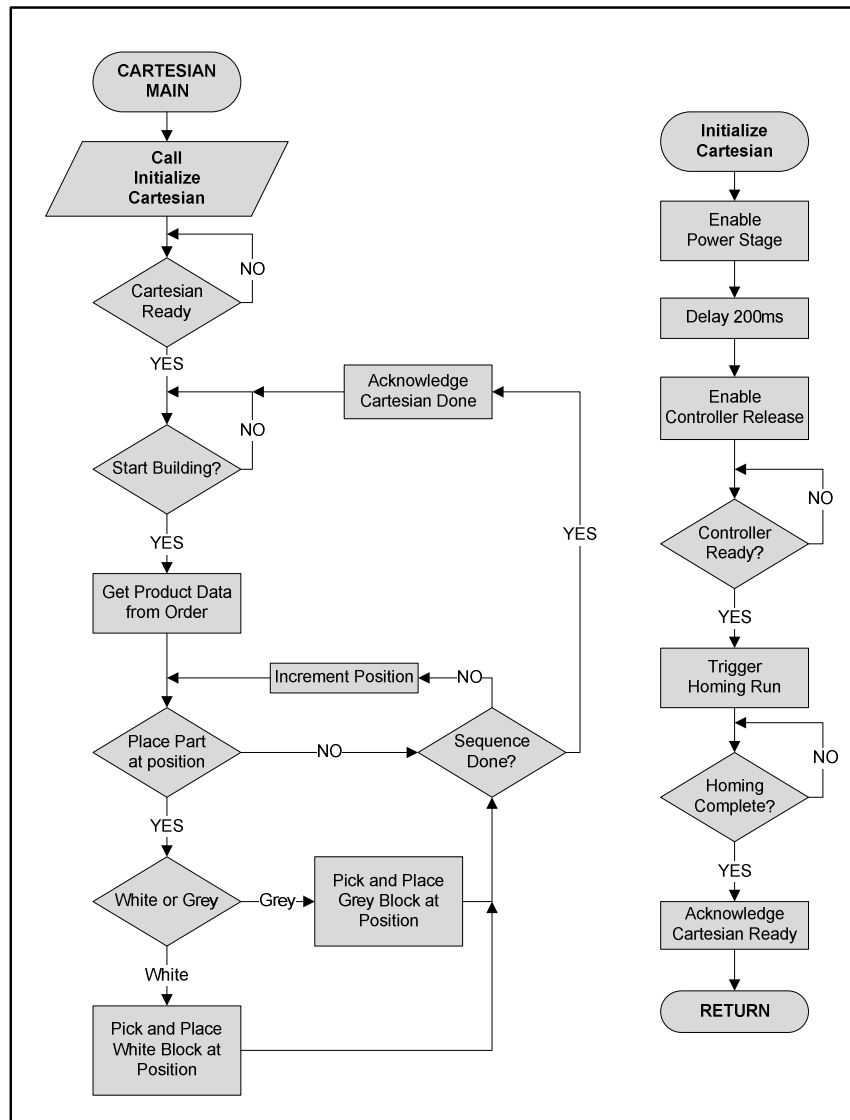


Figure 4.13 Cartesian controller algorithm

Once the Cartesian controller is instructed to start building a product, it starts executing the building sequence algorithm. The controller selects the first position on the twelve-position working tray (described in section 4.6.3.3), clears the position indicator register and tests if a block should be placed in that position or not. If no block should be placed, the indicator is incremented and the next

position is tested. Otherwise, the Cartesian controller tests which colour block (white or grey) should be placed, sets the corresponding data on the inputs of the motor controllers and triggers them to start moving. On motion complete, the Cartesian controller signals the actuators (cylinder and gripper) to pick up the part at that position. Next, the Cartesian controller changes the BCD value on the inputs of the motor controllers to the current position coordinates and triggers the motion. After motion complete, the actuators are signalled to place the part into the placing tray on the conveyor. Afterwards, the Cartesian controller tests if the product is finished or not. If it is not finished yet, the indicator is incremented to check the next position. Otherwise, if the product is finished, the Cartesian controller acknowledges “Cartesian Complete” to the process controller, and waits to be instructed to build the next product.

4.5.4 KUKA KR5 Sixx R850

Figure 4.14 shows the KUKA robot and its function in the system is to pick and place the remaining blocks into the twelve-position pallet to complete the products. It must either assemble two of the same products or two different products simultaneously, on the two conveyor lines in front of it (depending on priority and quantity). Furthermore, it is chosen because of its speed, extreme flexibility, agility and the repeating of tasks with high accuracy.



Figure 4.14 KUKA KR5 Sixx R850 articulated robot

4.5.4.1 Description

The KUKA robot is a 6-axis articulated robot, with a FESTO gripper mounted on its flange. Each axis is driven by a stepper motor, of which three of the axes manipulate the XYZ Cartesian coordinates and the other three manipulate the wrist actions (roll, pitch and yaw angles). Referring to Figure 4.15, it has a spherical work envelope with a reach radius of 855mm. Additionally, it can handle a payload of 5kg on its flange at full speed ($250^\circ/\text{s}$ or 7.6m/s) and have a repeatability accuracy of $\pm 0.03\text{mm}$ [80]. Moreover, it has pneumatic outputs switched by internal solenoids and inputs for feedback near the flange to operate EoA tooling. Furthermore, it has digital IOs situated on the back of the controller for miscellaneous signalling and supports protocols like Ethernet®, Profibus® and DeviceNet™.

Finally, safety sensory and perimeter guarding equipment must be connected to the safety contact at the back of the controller to enable the motor drives and enable operation of the robot.

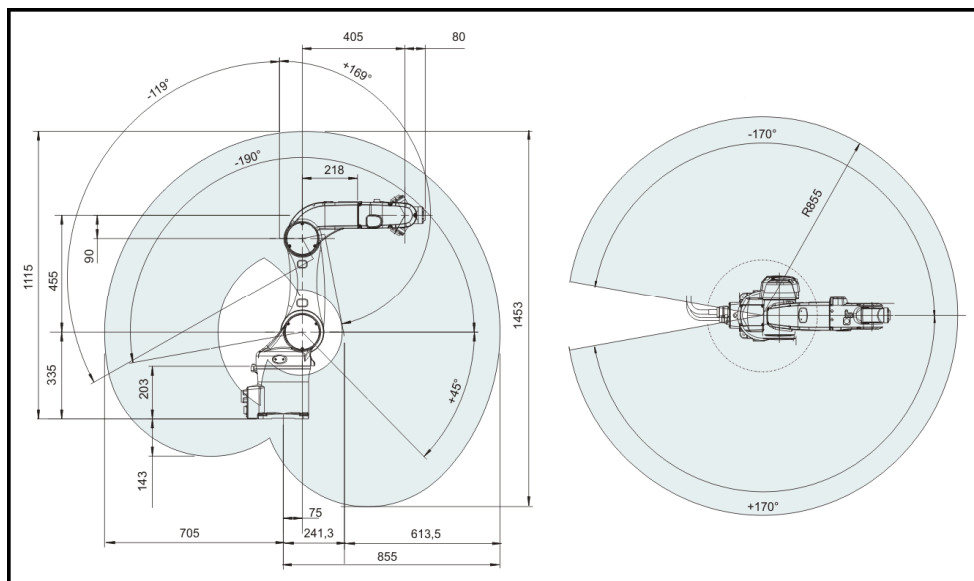


Figure 4.15 Work envelope specifications [81]

4.5.4.2 Programming and Calibration

Using the KUKA control panel (KCP) shown by Figure 4.16, an operator or programmer can calibrate tool and base profiles, control robot motions and program operations. Robots are normally taught motions and operations by

leading or jogging it through the desired movements it must follow and then saving those positions in the program of the robot. Alternatively, if the robot is correctly calibrated, a programmer can use the Cartesian coordinates or “set” and “turn” values to manipulate the motions of the robot. Table 4.1 provides a description of the labels in Figure 4.16.

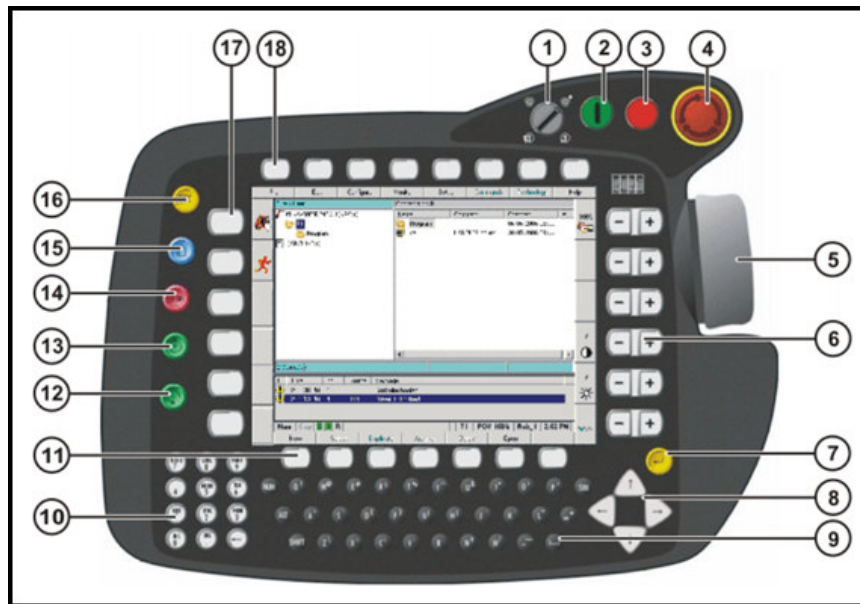


Figure 4.16 KUKA control panel [82]

Table 4.1 Description of KCP Labels [82]

Label	Description	Label	Description
1	Mode selector switch	10	Numeric keypad
2	Drives ON	11	Soft keys
3	Drives OFF	12	Start backwards key
4	Emergency Stop button	13	Start key
5	Space Mouse	14	STOP key
6	Right-hand status keys	15	Window selection key
7	Enter key	16	ESC key
8	Arrow keys	17	Left-hand status keys
9	Keypad	18	Menu keys

Firstly, the tool and base profiles can be calibrated. To calibrate a tool the 4-point method is used [83]. This involves that the tool centre point (TCP) is defined by teaching the robot four varying positions touching the same defined reference point with the TCP. The robot controller then uses the four positions to determine where the TCP is situated in the space in front of the flange. Next, the calibration of the base profile entails that the robot is jogged so that the TCP touches the zero position of the working platform to be taught as the base. Following this, the robot is jogged along a horizontal edge of the platform to calibrate the X-axis. Likewise, the robot is jogged along the adjacent edge (at a square angle respective to the X-axis) to calibrate the Y-axis. The controller uses the X and Y motions to determine the angle of tilt and orientation of the platform (a platform is not always level or at right angles with the robot).

When the tool and base profiles are calibrated, the robot can be programmed using KUKA Robot Language (KRL), which has a similar syntax to C-language. In addition, a programmer must be granted “expert” or “administrator” rights, to have full capability in manipulating the programs of the robot [83]. Additionally, an external keyboard, mouse and screen can be connected to the robot controller along with the KCP to further ease programming.

Firstly, by creating a new program, a template of a basic program which consists of a main loop and an initiating movement (homing run) is opened by default. The programmer can now teach motions by moving or jogging the robot to a desired position and then saving that position in the program memory. Various types of motion exist. Examples of these motions include point to point (PTP), linear (LIN) and circular (CIRC) motions. Moreover, elements like velocity, TCP profiles and end or transition conditions can be specified to fine-tune the behaviour of these motions.

Alternatively to jogging and teaching, the programmer can specify the same movement types as above, by using the coordinates (XYZABC) or “status and turn” values directly in the command. This enables this programmer to manipulate the robot motions by using functions and calculations instead of teaching. Using this method, the programmer must ensure using a tool profile with its corresponding base profile, otherwise results can be hazardous.

4.5.4.3 KUKA Control Software

The KUKA robot implements a similar algorithm to that of the Cartesian, but with minor differences. Firstly, the algorithm and all the logic to control the gripper run in the KRL program of the KUKA and not on the main PLC. The main PLC determined which product must be built, supply the KUKA with a corresponding program number and trigger it to start building. Afterwards the PLC waits for an acknowledgement from the KUKA to signal it is done.

Referring to Figure 4.17, when the system is powered up, an operator must do a manual homing run, to ensure that the robot returns to mechanical zero safely and without collisions (homing run is done manually due to safety). Afterwards, the operator can enable the drives and run the KUKA in external PLC mode. The KUKA will send a ready acknowledge to the Main PLC to verify it is ready for operation.

When the KUKA is triggered to start building a product, it uses the program number supplied by the PLC to obtain the product recipe and starts executing the building loop. Similarly to the Cartesian, it selects the first position on the twelve-position working tray, clears the position counter and starts the sequence.

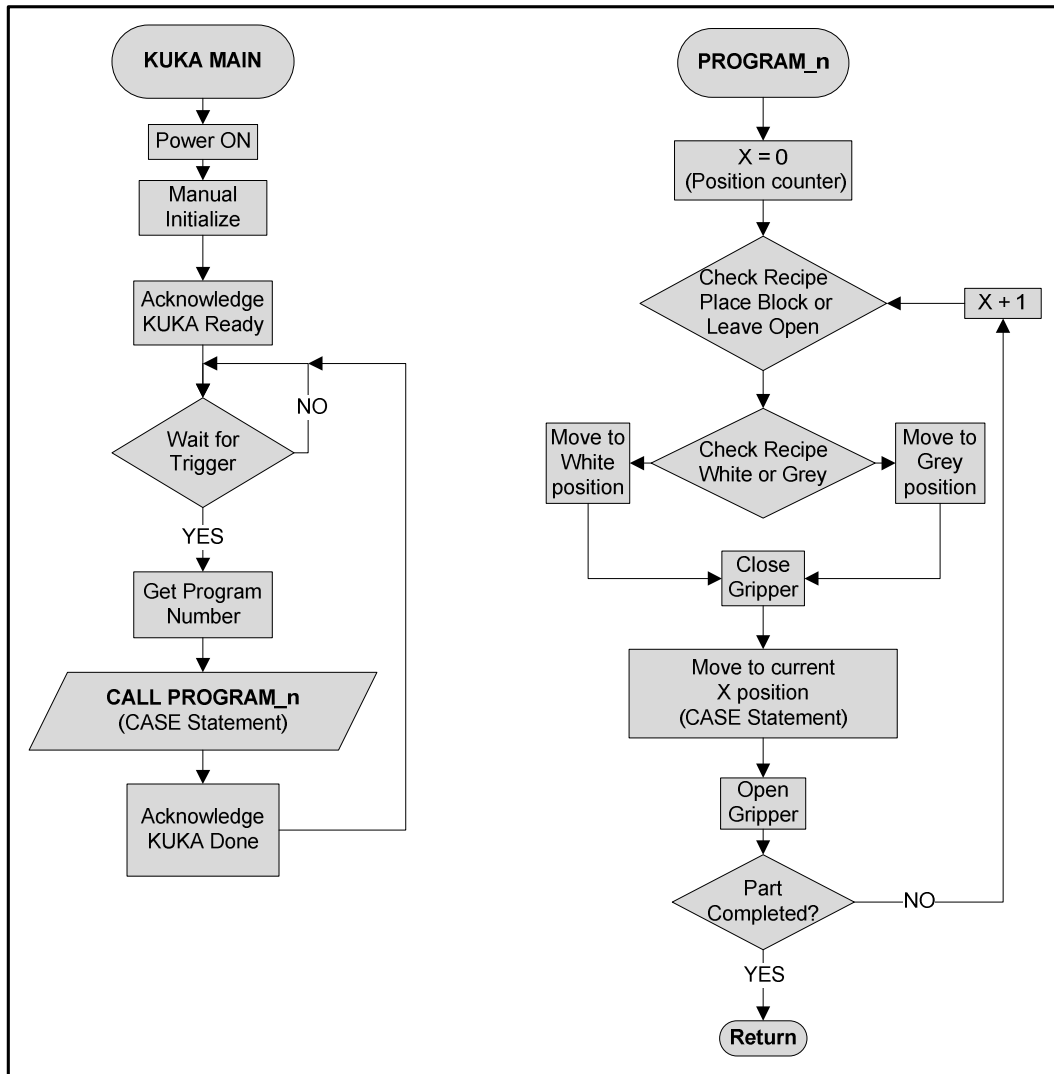


Figure 4.17 KUKA control program

The KUKA first tests if a block should be placed in the current position or not. If no block should be placed, the counter is incremented and the next position is tested. Otherwise the KUKA tests which colour block should be placed, moves to the XYZ coordinates of that colour block and closes the gripper. Next, it retrieves the coordinates of the current position, moves to that location on the twelve-position pallet and opens the gripper. Afterwards, the KUKA moves to a waiting or via point and tests if the sequence is done. If not done yet, the counter is incremented and the following position in the sequence is tested. Otherwise, like the Cartesian, if the sequence is done, it acknowledges it to the main PLC and waits to build the next product.

4.5.5 Gantry Crane

Firstly, the function of the gantry crane in the system is to pick completed products from the conveyor, then depending on pass or fail signals from the PLC, either rejects it, sends the product for rework or releases it for dispatch.

4.5.5.1 Description

Figure 4.18 shows the gantry as it is currently used in the system. It is a custom-built 2-axis (XZ) pick-and-place robot, which is controlled by a PLC and interfaced with the main controller using DeviceNet™. The X-axis consists of a linear drive, which is purely operated by pneumatic air and solenoids, and the Z-axis consists of a cylinder and a parallel gripper. In addition, a displacement encoder is mounted parallel to the linear drive. The displacement encoder is basically a potentiometer with the wiper connected to the sliding part of the drive. Furthermore, if the drive moves, the resistance value on the displacement encoder changes and this provides feedback to the PLC.

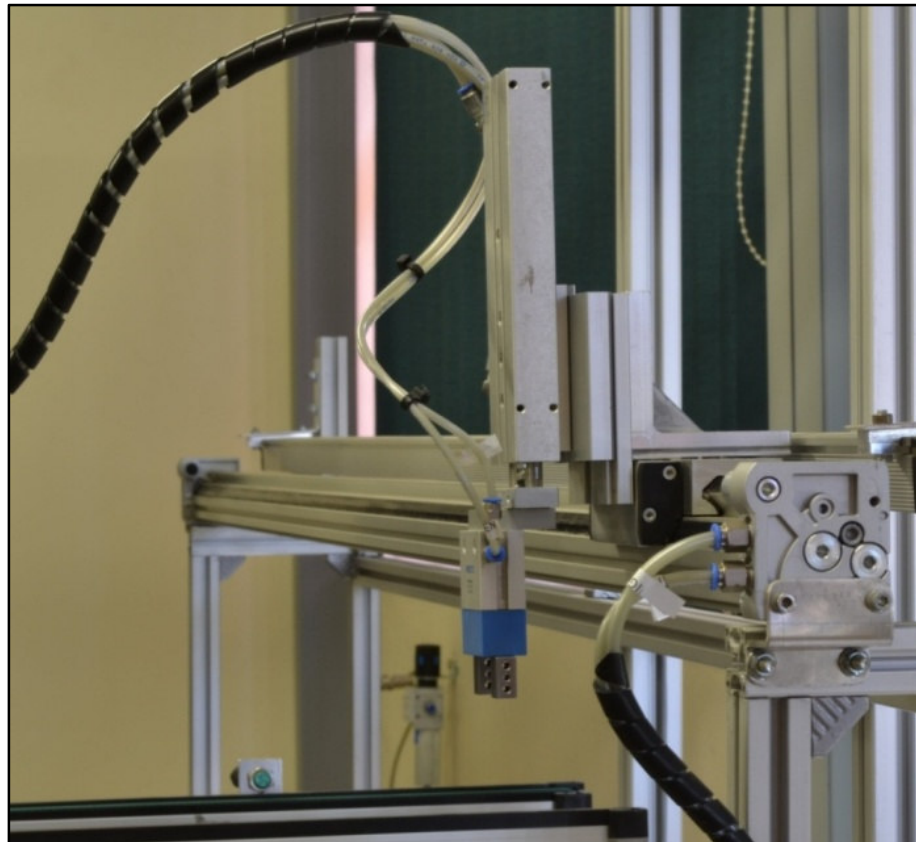


Figure 4.18 Gantry crane

4.5.5.2 Setup and Calibration

The procedure to teach and calibrate the reference positions of the gantry crane are as follows:

Firstly, an online connection is established between a programming terminal and the PLC. Furthermore, the real time monitoring and diagnostics is run to see the internal registers of the PLC change. Next, the slider part of the linear drive of the gantry is manually pushed into the desired position, the corresponding register value recorded and stored in the PLC memory. Finally, the PLC is reprogrammed and verified for correct operation.

4.5.5.3 Operation

To move to a defined position on the drive, the PLC retrieves the reference of the position it must move to, reads the encoder and compares it to the reference to determine which direction the drive must move. Afterwards, the PLC actuates the corresponding solenoid to start the motion and keeps comparing the current reading of the encoder to the reference. Once the encoder value is equal to the reference, the motion is stopped.

4.5.5.4 Gantry Programmable Logic Controller Operation

Referring to Figure 4.19, at power up the gantry initializes itself, which entails moving to the reject position and rejecting any “unknown” product that might still be in the gripper. After initializing, the gantry moves to the waiting position and sends a gantry ready acknowledgement to the main controller. When triggered by the main controller, the gantry tests for a pass or fail, then moves to the pick position, and signals the actuators (cylinder and gripper) to pick up the pallet. Afterwards, the gantry moves to the corresponding position and places the pallet. After placing the pallet, the gantry moves back to the waiting position and waits for the next trigger.

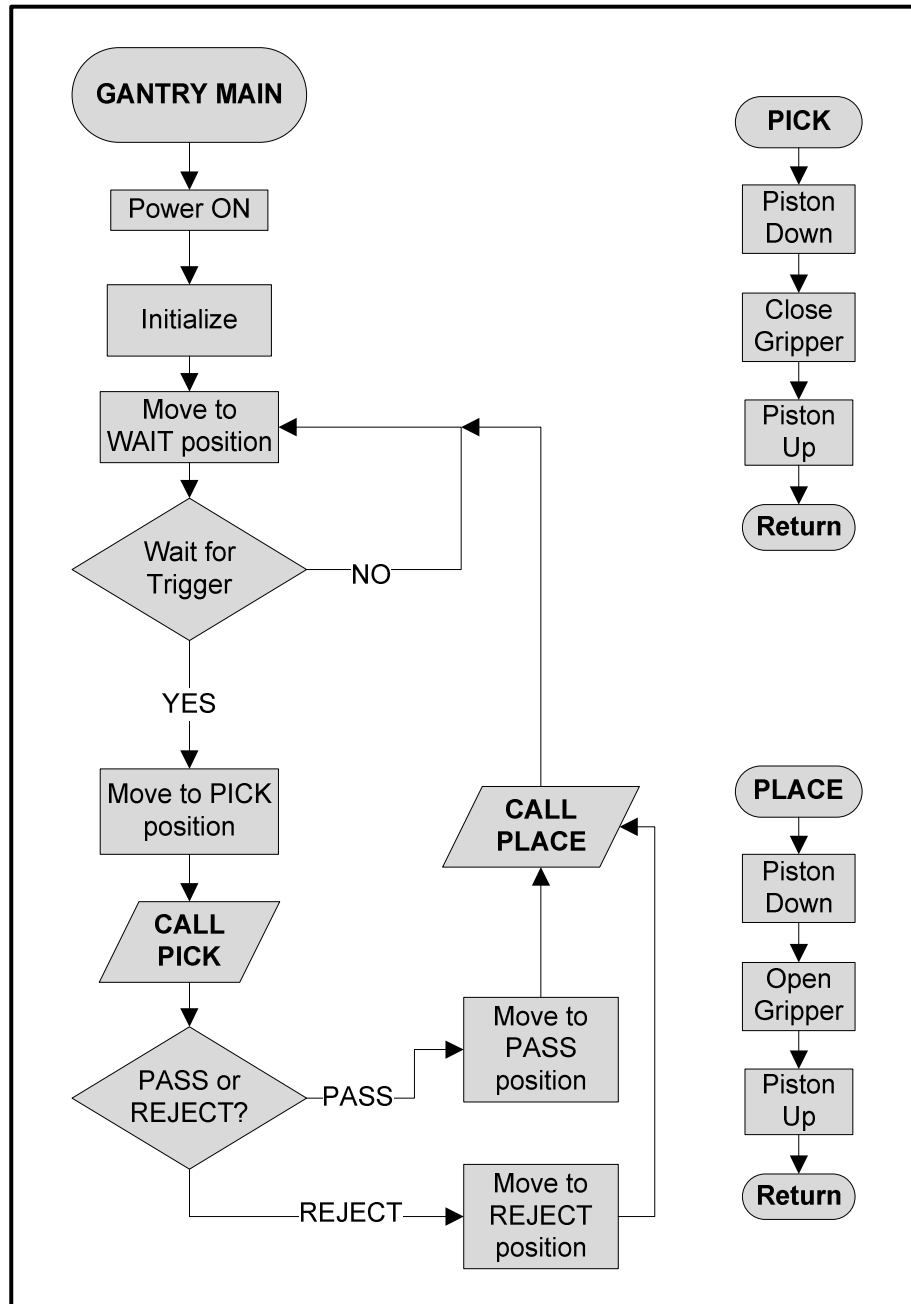


Figure 4.19 Gantry PLC program

4.5.6 Controller Hardware

Figure 4.20 shows the chosen Allen Bradley (AB) controller and can be programmed by using the ControlLogix 5000 programming software. It is built up using a ControlLogix 5563 processor, power supply, DeviceNet and Ethernet communication modules, two digital input modules and two digital output modules—which are all slotted into a 1756 chassis.



Figure 4.20 Allen Bradley controller [28]

In addition to controller components, Table 4.2 lists selected features included in the AB controller [28] [84].

Table 4.2 Controller features

Controller tasks	<ul style="list-style-type: none"> • 32 tasks • 100 programs per task • Event tasks: all event triggers
Built-in communication ports	<ul style="list-style-type: none"> • 1 RS-232 serial port
Communication options	<ul style="list-style-type: none"> • EtherNet IP • ControlNet • DeviceNet • Data Highway Plus • Remote IO • Third-party process and device networks
Controller connections supported	<ul style="list-style-type: none"> • 250
Programming languages	<ul style="list-style-type: none"> • Relay ladder • Structured text • Function block • SFC

4.5.7 PanelView™ Plus 700 Human Machine Interface

Figure 4.21 shows the chosen PanelView™ Plus 700 HMI from Allen Bradley to allow an operator to interface with the system [85]. This graphical interface has a 6.5 inch colour touch-screen display, with a 640 x 480 graphics resolution. The operator screens, running on the HMI, are developed using FactoryTalk® View. In addition, the HMI supports a keypad and twenty-two function keys for operator input and can be seen in Figure 4.21. Furthermore, it also has a RS-232, Ethernet and two USB ports available for communication. To conclude, this graphical interface offers a complete solution to interface between operator and machine.

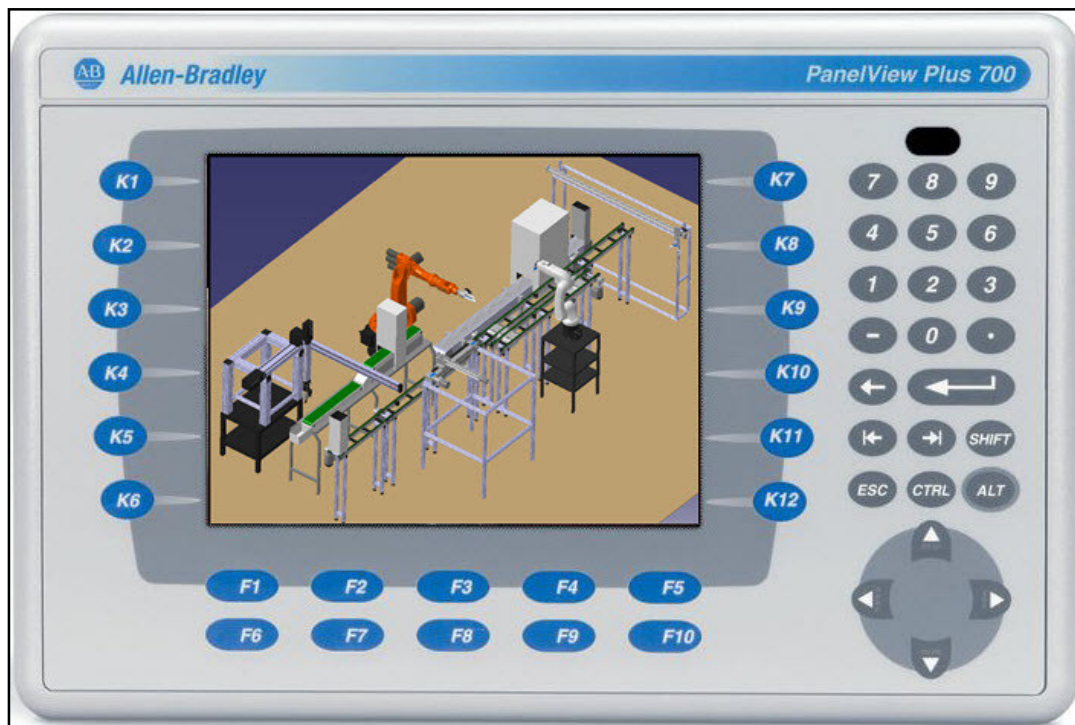


Figure 4.21 PanelView Plus 700 HMI

4.6 System Software Overview

Figure 4.22 shows the overall software structure of the system and how the different software modules and hardware components are interconnected. The proposed requirements of the system include that control must be alternated between different supervisory software instances, where each instance must perform production planning and oversee device and assembly operations. The

system consists of two software instances which must alternate control. The first being the PLC itself and the second a computer-based multi agent software system.

To achieve this, the PLC code is developed to execute several concurrent software modules. The main PLC routine utilizes a state machine to select between the control states (MAS or PLC), and the other specialized routines (handled subsequently). Each of these states enables certain functions of the system and excludes others, depending on the requirements of the selected state. In addition, a device handler to control assembly device operations executes concurrently with the state machine and is available for use by both the control instances in the system. Furthermore, additional subroutines and functions to aid the system are also available when requested by the state machine.

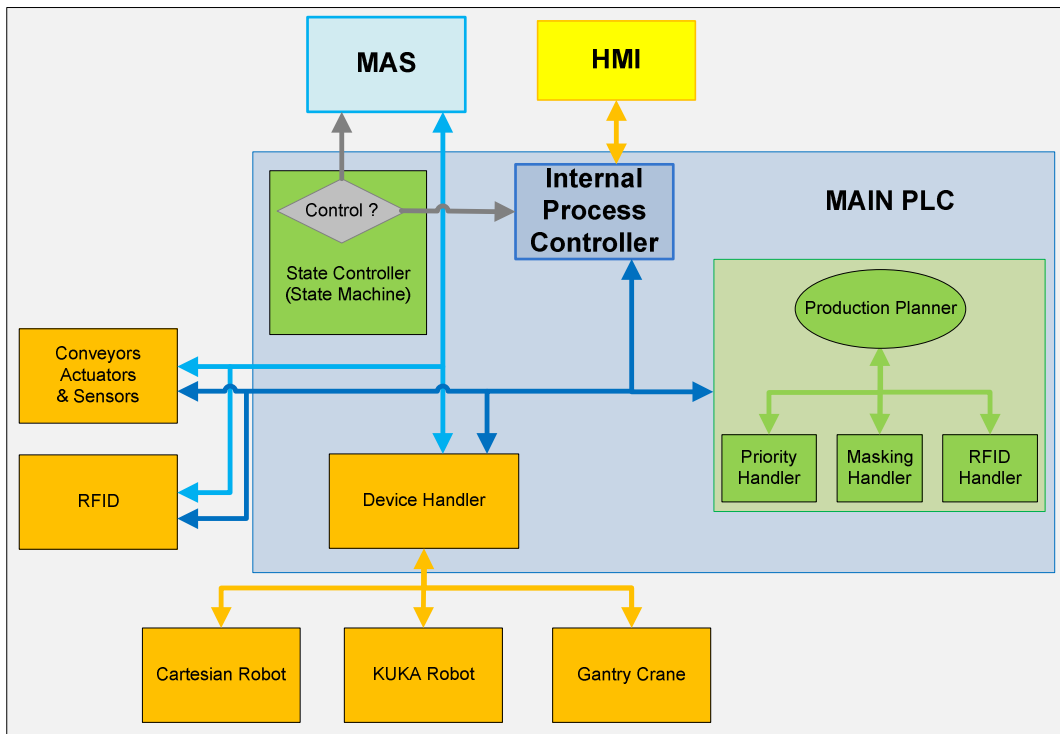


Figure 4.22 Software architecture

4.6.1 State Controller

Recalling from the preceding section, the main routine implemented in the PLC is a state machine and will subsequently be referred to as the state controller. The

state controller has different operating states which are defined as system self check, initialize, clear, internal process control and the MAS state. Each one of these states performs a certain function when selected.

How the state machine operates can be explained by referring to Figure 4.23. When power is applied, the system remains idle and the state controller switches to the “system self check” state. At this point the main PLC has full control of (over) the system and starts the system pre-checks, which include testing the air pressure, communications and if the KUKA robot is initialized (initialized manually due to safety). After the pre-checks are completed, the state controller waits for further instructions from an operator.

Upon instruction of an operator, the system starts up and the PLC tests if the system components (Cartesian and gantry) must be initialized. If initializing is necessary, the state controller changes state to “initialize”, where it instructs the Cartesian robot and the gantry crane to initialize and acknowledge when complete. After initialization, the state controller changes state to “clear”. On the contrary, if initialization was not necessary, the state controller will skip the “initialize” state and change directly to the “clear” state.

Once in the “clear” state, the PLC will start the conveyors and run the clearing routine. This will clear the system from any unknown, faulty and half-built products. In addition, clearing is also done when the system is in an unknown state or is about to perform a handover to another process controller. After clearing, the state controller will determine which one of the process controllers must seize control and change to that corresponding state.

Additionally, for one of the process controllers to acquire control, either one of the instances must first request for control and then wait for the instance having control to finish all its current processes. If the instance in control verifies to the state controller that processing is complete, the state controller will switch to the clear state, and afterwards hand over control to the requiring instance.

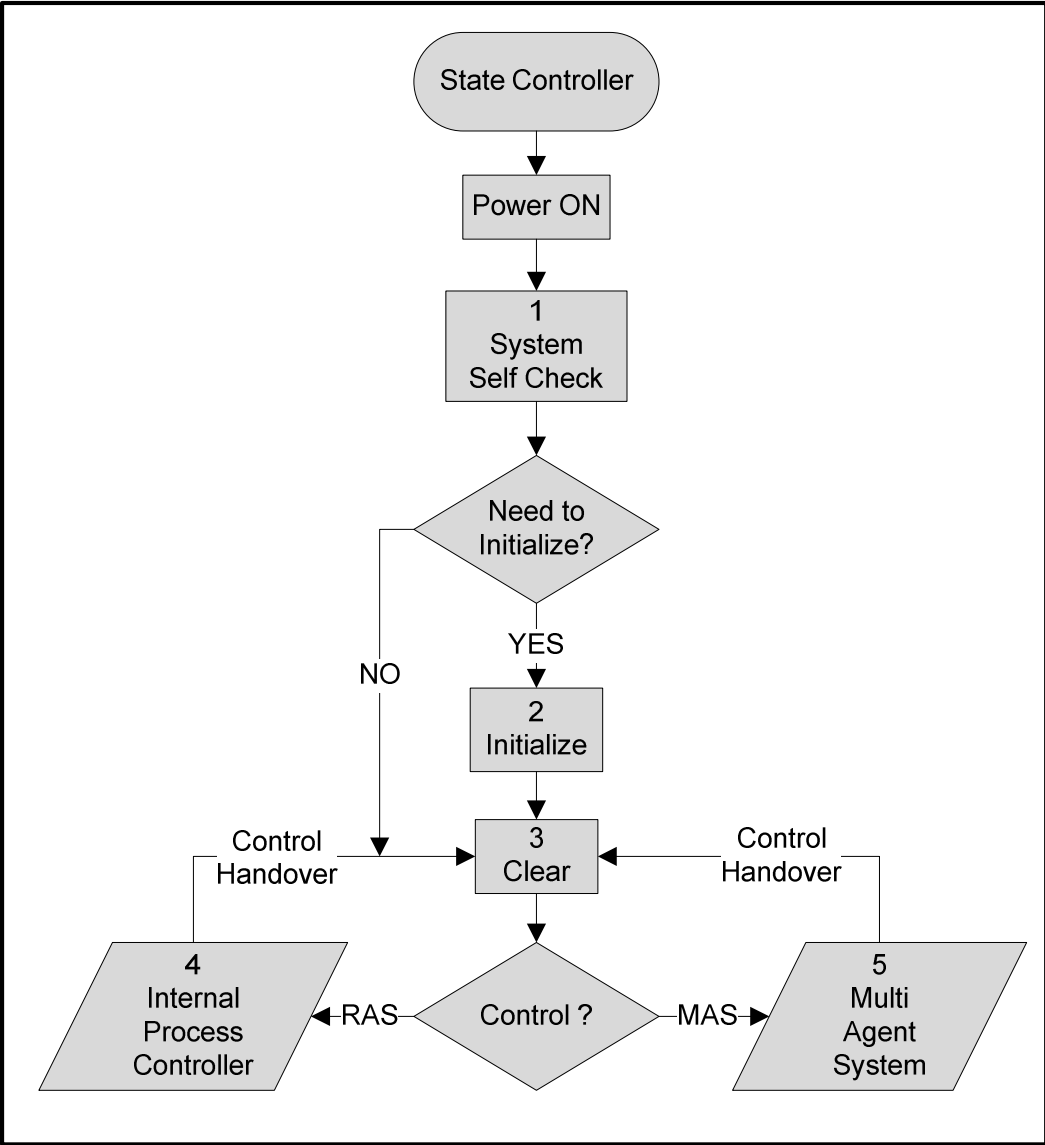


Figure 4.23 State machine utilized in state controller

4.6.2 Device Handler

The device handler is a control subroutine which resides inside the main PLC program to coordinate the actions that each assembly device must perform. For explanation refer to Figure 4.24. Firstly, the process controller monitors if a pallet is at a defined position on the conveyor. When a pallet is present, the process controller uses the production planner (described in section 4.6.3.2) to determine the build data and provides it to the device handler (for the gantry pass or fail is sent). Afterwards, the device handler then interprets the data and uses the build

algorithm of the corresponding device to instruct the device to perform the sequence of actions needed to build the product. After the building sequence is complete, the device handler acknowledges to the process controller that the device has finished building. This procedure runs concurrently for each device in the system.

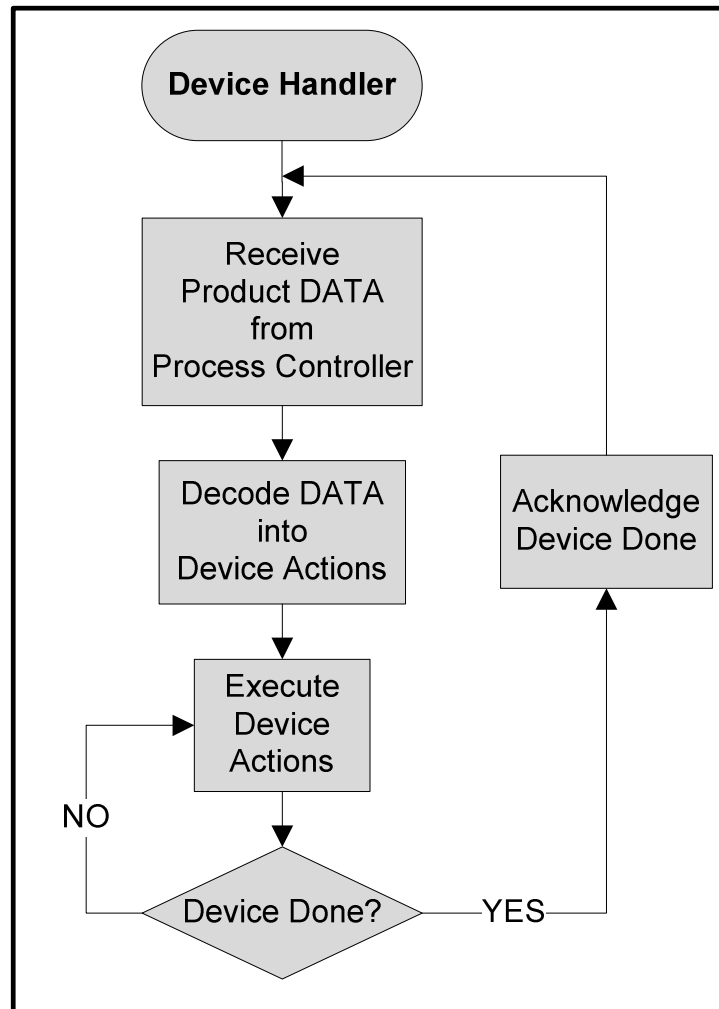


Figure 4.24 Device handler process

4.6.3 Internal Process Controller

Figure 4.25 shows the software functions utilized when the system is in the "internal process controller" state. The system accepts product orders from an operator panel (HMI), runs the internal production planner and also accesses the device handler to control the assembly devices. Additionally, the internal process controller also executes a subprocess to handle the conveyor actions. The

purpose of each software function used by the internal process controller is explained in the sections following directly.

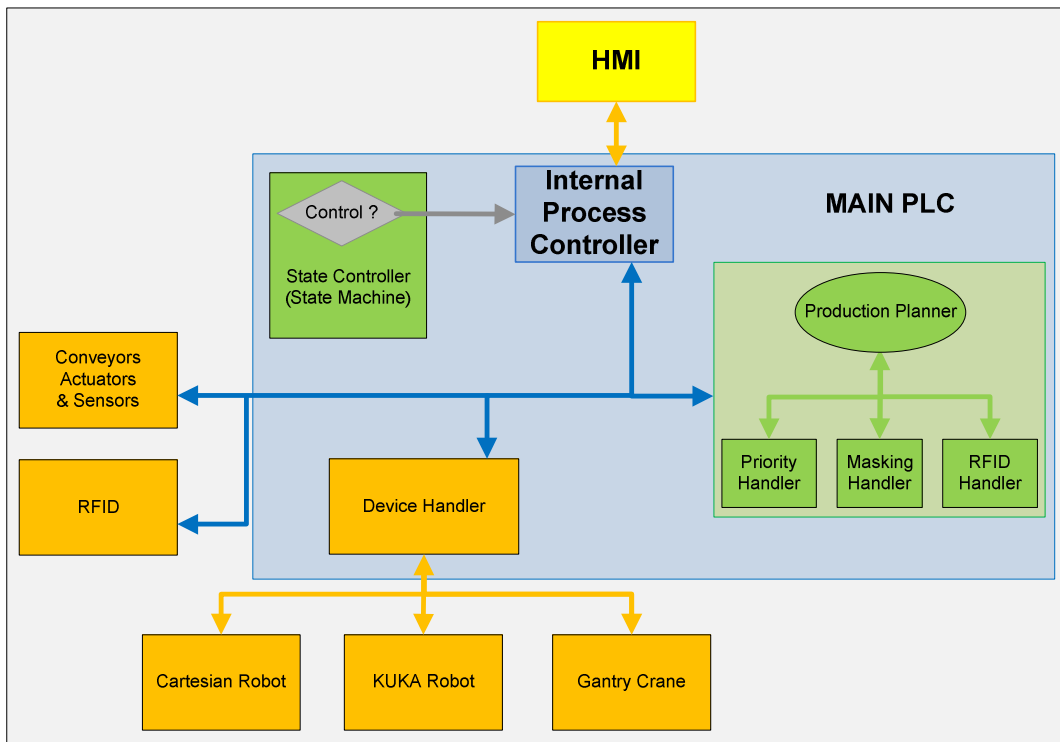


Figure 4.25 Internal process controller state

4.6.3.1 Interfacing with the System

The PanelView Plus 700 HMI (which was discussed in section 4.5.7) is utilized by the system to receive production orders from an operator. For subsequent explanations, only the internal process controller state will be considered.

After reaching the internal process controller state successfully, the interface will start up with a menu screen with various options. Available on the menu are options like system analysis, production history, new product creation and product ordering. Only product creation and ordering will be handled for this project.

When an operator desires to create a new product, the product creation screen is entered by selecting it from the main menu screen. Under product creation, the operator is able to select an arrangement of parts to create a customized product. These customized products are then saved for future use and later used in

default ordering. To prove the concept of building multiple products, three product variations will be built.

In addition to product creation, the product ordering screen is used to provide the assembly system with the necessary production data and instruct it to start or stop production. Moreover, orders can be made by three different methods. Firstly, the system is instructed to build a product by acquiring a product string (product strings is handled section 4.6.3.2) by reading a RFID which is included on each pallet. Here the system will require quantity from an operator and build products according to the information stored in the RFIDs of each pallet. Secondly, orders can be made by manually entering a product string. The operator is requested to provide the system with the quantity and priority of each product to be built. A drawback to using this method is, however, that the operator must take great care when determining the product strings and entering it afterwards. To ensure correct building of products, this method is not recommended and is only used for testing purposes. Lastly, orders can be made by selecting from the default pre-saved products (created using product creation) available to the system. Here the operator only needs to provide quantity and priority to the system. In addition, to keep track of the products on the conveyor, the latter two methods assign a product to be built to the RFID present on the pallet. Here the data on the RFID only identify the pallet, instead of providing building data. For the purpose of further explanation, only the default ordering will be considered.

Once the default order screen shown in Figure 4.26 is selected, an operator is able to provide the system with the necessary production data. Present in the order screen are the default products which the operator can choose from, as well as a provision to set the quantity and priority of each product to be built. To vary the quantity and priority, the operator must click on the plus and minus buttons under each label next to the corresponding product to change it. The values will update and display on the HMI.

In addition to quantity and priority, control buttons to start and stop production, navigate back to the main menu and a software E-stop are also provided in the order screen. The software E-stop was included for testing and simulation purposes and must not be used instead of the physical system E-stop. To start

production, orders are placed by altering the quantity and priority of each product, and then the operator has the option of starting production by clicking on the start button. At this point in time, the system stores the data obtained from the HMI for further manipulation to determine the production plan for each product (explained subsequently). Afterwards, the system uses the manipulated data to start building the products and returns to an idle state when building is complete. In case production must be stopped prematurely, the operator can click the stop button, and terminate production with the result that the system returns to an idle state. The operator is now free to place a new order or navigate to other system options.

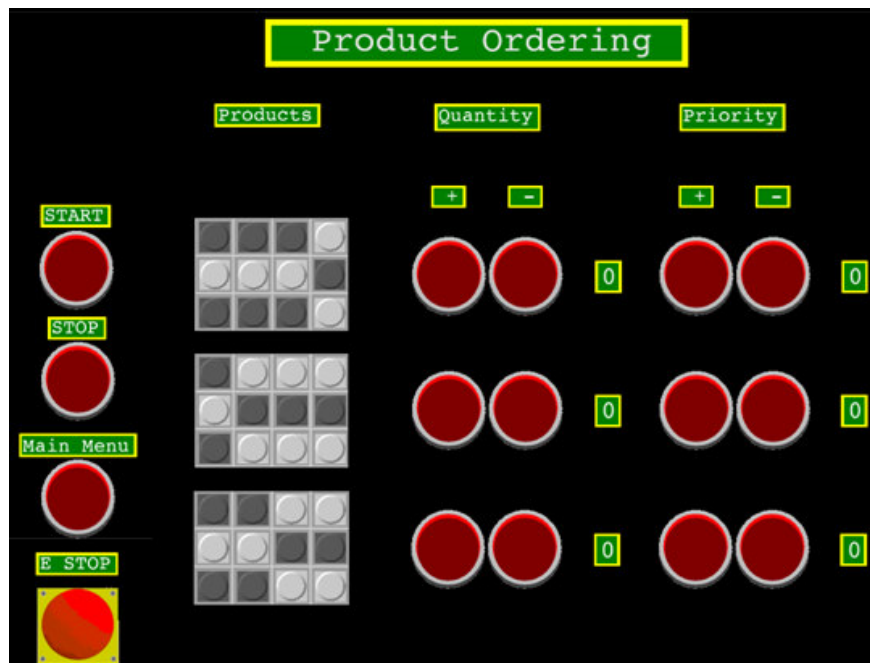


Figure 4.26 Product ordering screen

4.6.3.2 Production Planner

The production planner is a subprocess only available to the internal process controller and is implemented to supervise over the masking, priority and RFID handlers, as seen in Figure 4.27. The function of the production planner is to capture product orders, manipulate it into a building recipe, and then convey this recipe to the device handler to administer assembly operations. How the production planner determines building recipes and how the different handlers which it oversees operate are explained in the following sections respectively.

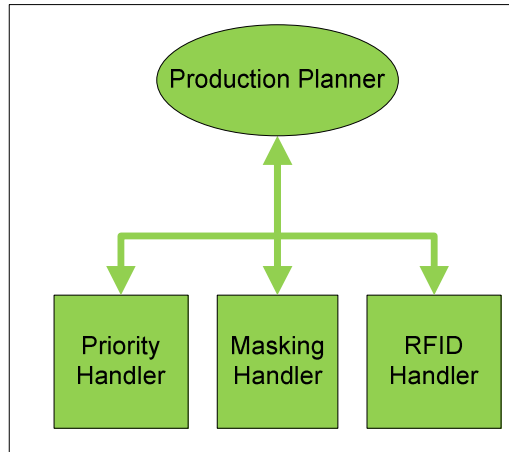


Figure 4.27 Production planner

4.6.3.3 Twelve-Position Product Tray

To demonstrate the concept that the RAS can build multiple products on the same conveyor line, a twelve-position product tray, shown in Figure 4.28, is used to assemble variations of a product in the same product family. The product tray has twelve slots, in which white or grey blocks can be placed to represent different product variations. Furthermore, each slot in the tray is allocated a reference number, as seen in Figure 4.28, and is then used by the device handler to decode the product strings into device actions. To rearticulate, each reference position or slot located on the product tray represents a location where a part can be placed.

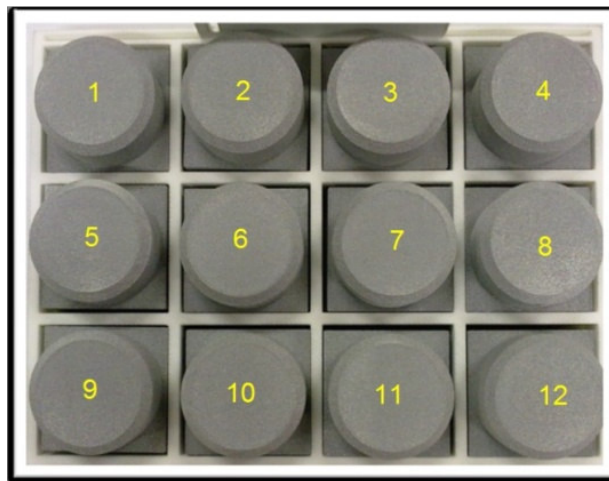


Figure 4.28 Twelve-position product tray

4.6.3.4 Product Strings

With reference to section 4.6.3.1, product strings are obtained by ordering products or entering them manually using the system HMI and storing it to the PLC memory. These product strings provide the system with the necessary information to build related products and are later manipulated to determine the building recipe for each product at different assembly cells.

Figure 4.29 shows an example of a twenty-four bit product string. Moreover, each bit in the string represents one of the twelve reference locations which are allocated to the product tray. Furthermore, a string contains two components of information. The first part contains block placement data and the second part block colour data. Block placement data includes information as to whether a block should be placed at a reference location or not. On the contrary, if a block should be placed at a reference location, the colour data provides the colour of the block to be placed. This enables the system to interpret where in the pallet a block should be placed, and which colour at this specific position.

In addition, the system uses these strings to determine which part of the product is built at which assembly cell. This product data are used extensively by the Cartesian and KUKA robots, and is explained in the succeeding section.

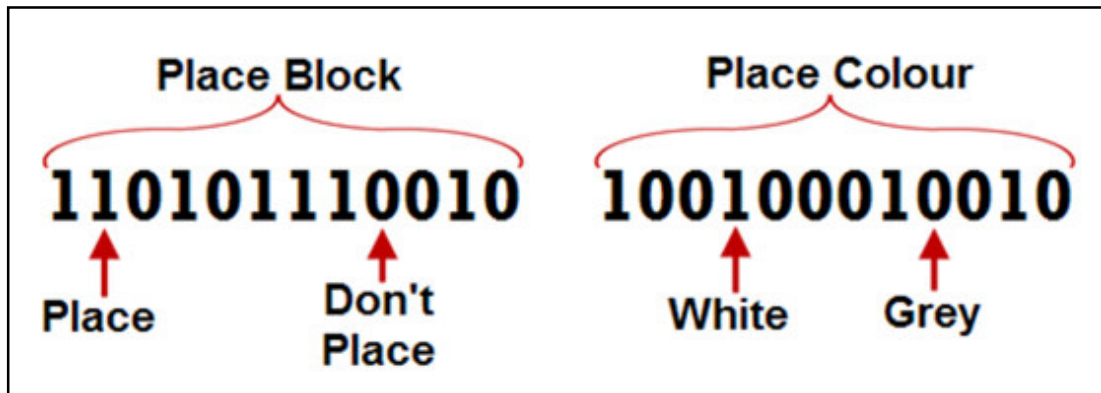


Figure 4.29 Example of a product string

4.6.3.5 Product String Manipulation (Masking Handler)

Recalling from section 4.3, the Cartesian robot is assigned to build all the matching features between products, and the KUKA robot builds the remaining features. To achieve this, the product strings obtained from product ordering must

be manipulated in some way to determine a production plan for the ordered batch. To achieve this, the production planner executes the masking handler subroutine to enable the internal process controller to determine where which part of a product is built.

To explain how the software routine functions, it can be represented by the logic equivalent circuit shown in Figure 4.30. Firstly, the respective product strings are bitwise AND together and the result stored in a temporary location in the PLC memory. At the same time, these product strings are bitwise NOR together and the result stored in a different memory location. Afterwards, the two results in the PLC memory are bitwise OR together to find all the matching bits between the respective product strings. The result obtained from the bitwise OR is used as the Cartesian masking string. In contrast, the KUKA masking string is obtained by taking the complement of the Cartesian masking string.

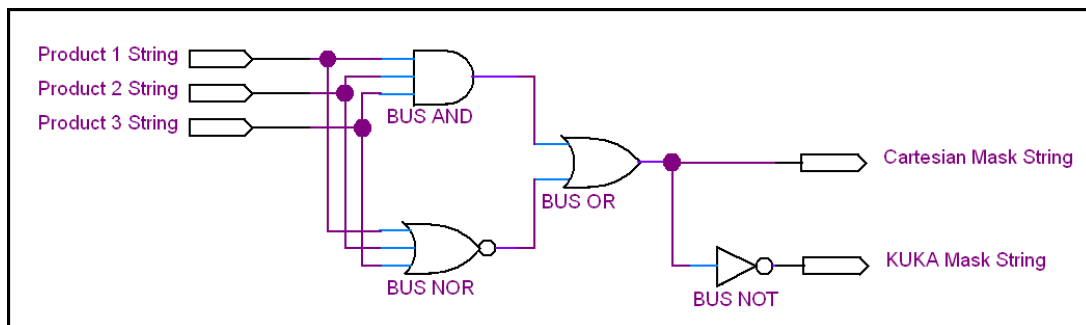


Figure 4.30 Logic representation of obtaining masking strings

To illustrate how the product masking strings are obtained can be better explained by referring to Figure 4.31. Firstly, orders are made using the system HMI to obtain the respective product strings along with their quantities and priorities. The three desired products are visualized at number 1. Afterwards, the masking handler compares the product strings to find matching features between the products. This is shown at numbers 2 and 3. By obtaining the matching features, the masking handler determines which part of the product should be built at which assembly robot. The resultant of what the KUKA builds and what the Cartesian builds are shown at number 4. In addition to finding the masking strings, the system must also determine which product must be built on which parallel conveyor line at the KUKA robot. By referring to section 4.5.1, this is

simply achieved by building the product with higher priority on lane A (strait) and the other on lane B (branch).

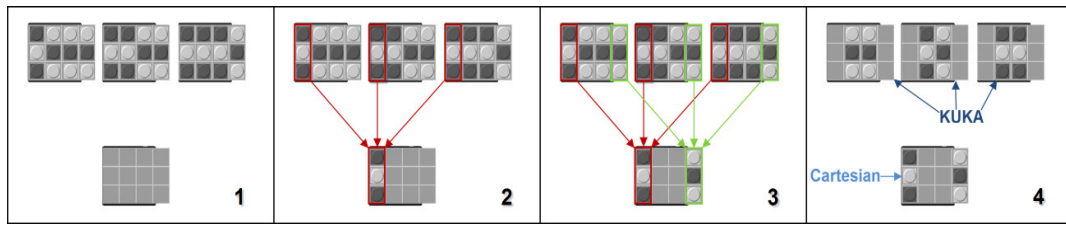


Figure 4.31 Comparing and masking scheme

4.6.3.6 Priority Handler

In contrast to the masking handler, the priority handler establishes the order in which products are built and decides what should be built at the parallel conveyor lines at the KUKA robot. To clarify, it must determine if different products must be built on each line or if the same product must be built on both lines. Initially, the building order of products is prioritized by assigning each product with a priority value between one and five when ordering. Value one being the highest priority and five the lowest. To clarify, if two different products have the same priority, then the product which is scanned first (higher on scan list), has precedence.

For explanation of the routine refer to Figure 4.32. Firstly, the priority handler sets the system priority equal to one. Afterwards, the priority of each product is scanned until the priority handler finds a match. If it finds a match, the quantity of the corresponding product is tested and the products are built until the quantity reaches zero. If the quantity is zero, the priority handler continues to scan for other products with the same priority. Afterwards, if no more products with a matching priority and a quantity more than zero are found, then the system priority is decreased to build the lower priority products. The priority handler repeats the above procedure until all the priority values are scanned and the quantities for the matching products are equal to zero.

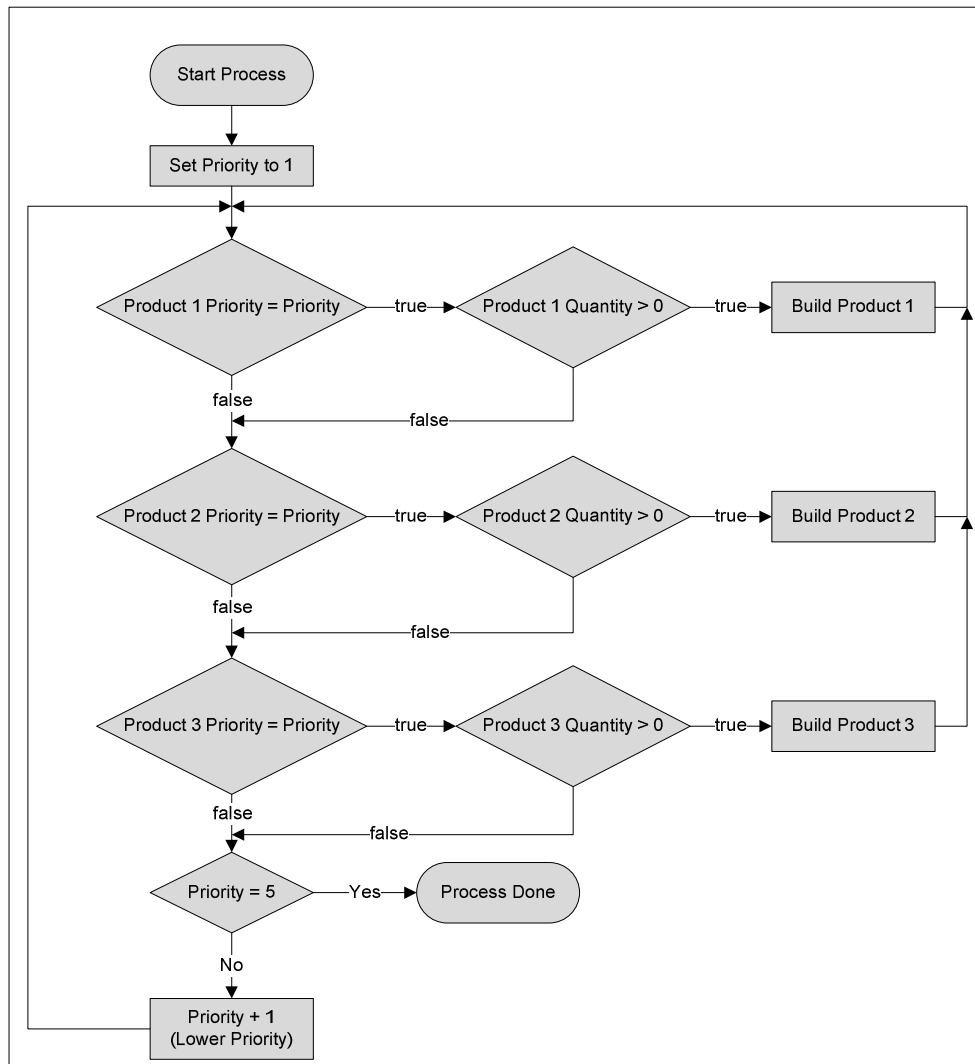


Figure 4.32 Priority handler routine

4.6.3.7 Radio Frequency Identification Handler

The function of the RFID handler in the system is simply to monitor all the RFID readers installed at various locations on the conveyor. In addition, when a tag is detected by a reader, the handler stores the tag data and which reader detected it to determine its location. Afterwards, this data are relayed to the production planner to track the products present in the system.

4.6.4 Multi Agent System State

A study completed by the RGEMS research group included “the development of process control and configuration in a reconfigurable production system using a

multi agent software system” [86]. A software agent is a computer program which works autonomously and goal-driven in a dynamic environment on behalf of a human or any other computational entity, possibly over an extended period of time, without direct supervision, to transform goals into executable tasks. In addition, a multi agent system is a community of software agents, working together to achieve a common goal.

Figure 4.33 illustrates how system components are interconnected when the state controller activates the MAS state. In this state, the MAS seizes control over the PLC functions, as well as selected software modules like the device handler. To clarify, the MAS is responsible for process planning and control; and instructs the PLC to do operations by monitoring and updating PLC tags in the PLC memory through OPC. In addition, however, the MAS cannot ensure accurate and stable delay timing, which is used to trigger stop-gates and operate the lift transverse conveyors; therefore, tasks that require accurate timing are allocated to the PLC.

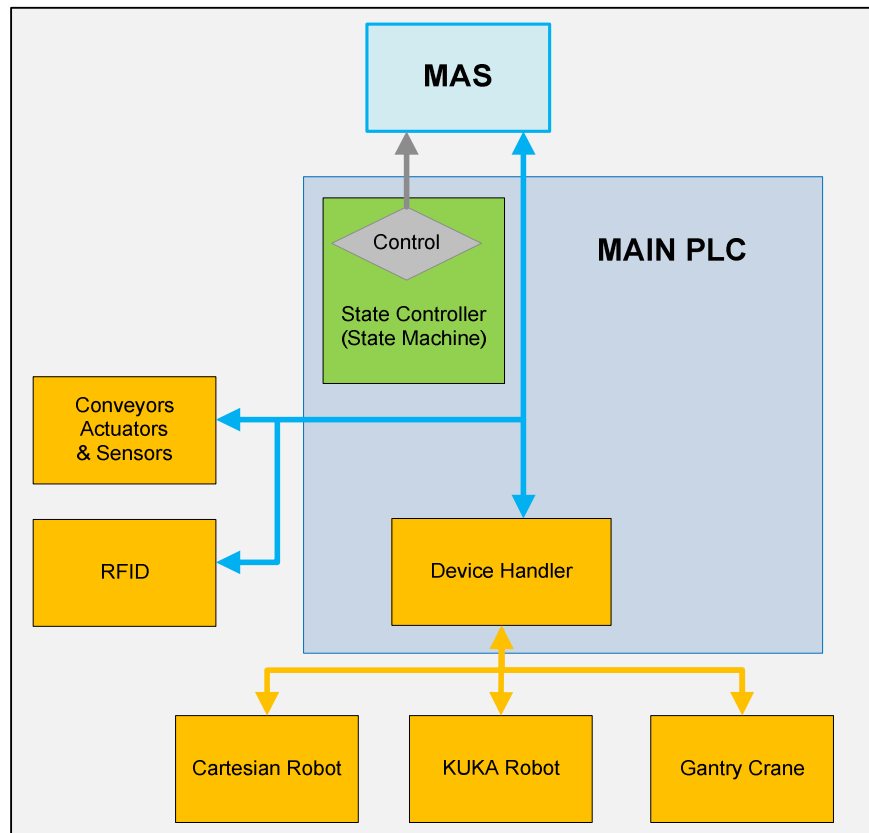


Figure 4.33 MAS state

Furthermore, since the MAS is responsible for its own production planning, it monitors the proximity sensors present on the inputs of the PLC and, based on their states, executes its production plan by triggering the corresponding conveyor and device actions. To control the assembly devices, the MAS sends the determined production data to the device handler in the PLC, where the device handler takes over the building process of the products, and relays the required instructions to the assembly devices. After a device has finished assembling, the device handler returns an acknowledgement to inform the MAS that the device operations are complete. This procedure continues until either the production has stopped or the MAS hands back control to the PLC.

4.7 Safety Precautions Implemented in the System

The assembly system developed in the RGEMS research laboratory utilizes multiple safeguards to ensure the safety of laboratory personnel. The machinery and conveyors are enclosed by a structure built from aluminium profile. The structure provides an overhead cable tray, where the wiring used for the equipment can be situated and protected from damage. Furthermore, the frame acts as a basic perimeter fence, preventing observers from wandering in proximity of dangerous machines while it is operating. Paired with the structure, infrared curtains are mounted on the exterior (envelope) of the frame. The curtains form a virtual barrier and provide feedback signals to the main power control circuit. In addition to securing the perimeter, the system is fitted with multiple E-stop switches, which are situated at strategic positions. The E-stop switches are connected in combination with the IR curtains and are hard-wired to the power control circuit. The power circuit entails that the AC-mains are wired in series with two safety relays. The function of the relays is to interrupt the AC-mains supply, if one of the safety precautions is breached. The usage of two safety relays ensures that the system power is reliably interrupted, regardless of whether one of the relays fail due to its contacts “welding” together (current overload).

Finally, a pressure-sensing safety mat is installed in a safe area outside the KUKA robot’s work envelope (reach). The safety mat ensures that a technician can safely program or calibrate the robot while standing on top of the mat,

including that the robot must be forced to move at a slower speed. The safety mat is wired to the robot controller and acts like an E-stop switch during programming. With these safety measures in place, the safety of laboratory personnel and observers present in the vicinity becomes apparent.

4.8 Conclusion

This chapter focused on the specific requirements in hardware and software components, how they were chosen, and the development of the required methods to implement them to accomplish the development of a RAS. A brief overview of the intended system layout and hierarchy, operation of the system and product flow was discussed. Furthermore, the various assembly devices in the system were described, along with their operation, capabilities and the methods to control them. In addition to assembly devices, the overhead software to capture orders from an operator, determine and control production processes and relay instructions to the assembly devices were also considered. Moreover, a brief introduction and background information on OPC and KEPServer were provided, followed by its setup and use as a flexible communication bridge between the implemented devices from various different vendors and the control elements. Furthermore, some of the implemented hardware devices can be potentially dangerous, which is why precautionary safety equipment were implemented, preventing damage to devices and ensuring safety of laboratory personnel.

In conclusion, by utilizing the hardware and software components chosen in this chapter, and implementing the overhead software with the embedded functionality, planning methods and communications, a system with typical characteristics of a RAS, which consists of flexible reusable industry standard components that are used in typical industry applications, and comprises the capability to easily reconfigure or recalibrate with minimum effort from an operator to limit downtime, can be attained.

Chapter 5 Testing and Results

5.1 Introduction

This chapter reports on a series of tests that were undertaken to verify the operation of the system and its subcomponents, by initially making use of simulation and afterwards verifying the results through physical testing.

The tests are compiled in such a way that the fundamental functions are verified first, and then afterwards used to verify system subprocesses, and eventually testing the system as a unit. The testing includes verifying each assembly device separately while investigating the three verification methods and distributing the project objective amongst each test. Furthermore, the testing is also separated into verifying the system with PLC in control as well as MAS in control.

The procedure in which testing was conducted as well as the subsequent section can better be explained by referring to Figure 5.1. In Figure 5.1, notice that the blue-coloured blocks represents the start of each test to be conducted. In addition, the grey-coloured blocks represent what is tested along with the methods of testing, and lastly the green-coloured blocks show the sections of the system that has been verified by testing.

Initially, test 1 shows the verification of the production planner which is used in both the internal process controller of the PLC and in the DELMIA simulations. Test 1 includes testing the priority and masking routines. Next in test 2, each assembly device is verified using a respective verification method along with certain system capabilities. The first method verifies the KUKA robot using a process plan. The second method verifies the gantry crane with a virtual PLC. Lastly, the third method verifies the Cartesian robot using the actual system PLC. This is known as virtual commissioning. In contrast with tests 1 and 2, test 3 verifies the system with the MAS in control. It uses the physically built system which is verified by tests 1 and 2 to perform a simple assembly operation. As a result the entire system is verified with both PLC and MAS in control.

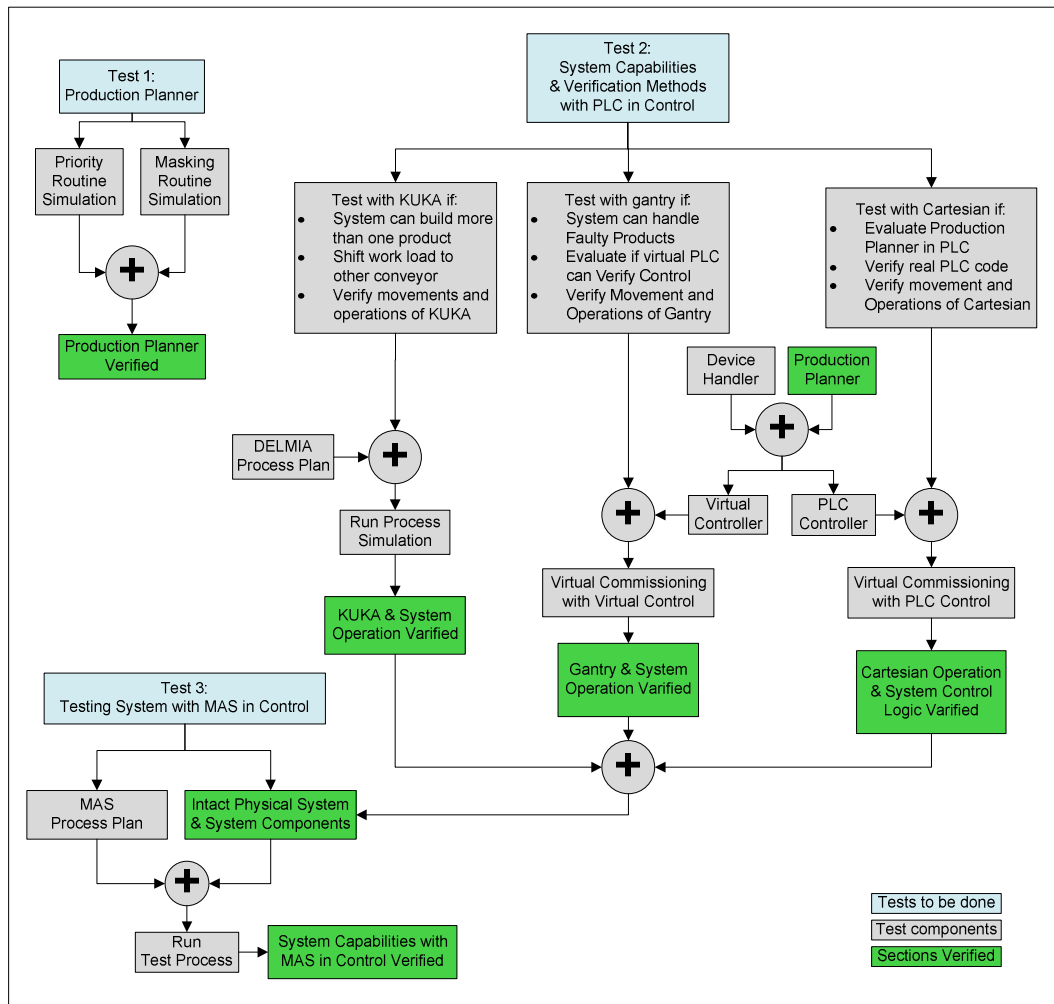


Figure 5.1 Testing to verify system

5.1.1 System layout

Figure 5.2 shows the physical layout of the system components, as currently implemented in the RGEMS laboratory. Furthermore, it shows the physical wiring which contains power connections, sensor wires and output actuators. Moreover, it is also identified where all the control logic devices are allocated and how these are interconnected. For a detailed description of the system layout, also refer to section 4.2.

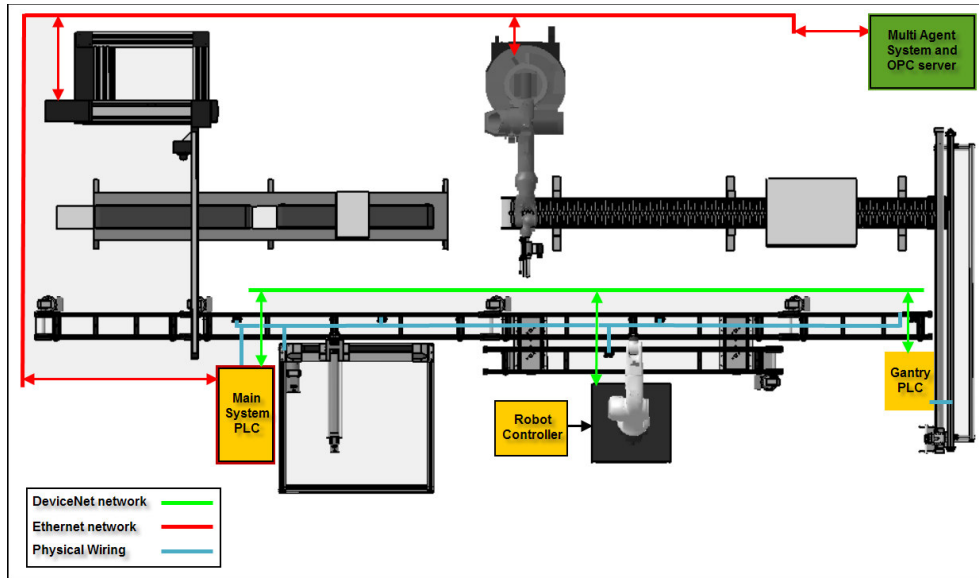


Figure 5.2 System layout

5.2 Tests Done in Internal Process Controller Mode

Figure 5.3 shows all the active processes and functions in the PLC when the internal process controller is enabled.

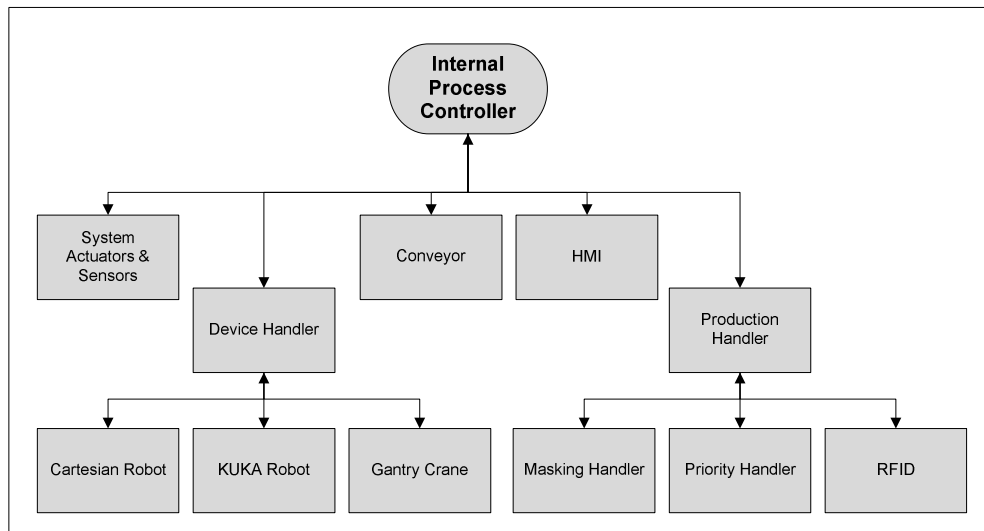


Figure 5.3 Active processes in PLC mode

5.2.1 Testing the Production Planner

To verify the production planner which supervises over the masking, priority and RFID handler routines, each handler was tested separately. The RFID handler

simply returned an identification value when a RFID tag is in proximity of a reader and stores the value in a location the PLC can use, and thus will not be tested. On the contrary, the masking and priority handlers were loaded into a simulation environment, whereafter simulations were conducted to obtain results. These simulations and the results thereof are revealed in the succeeding sections.

5.2.1.1 Masking Handler

Table 5.1 shows the setup and the results obtained from a simulation done to model the output of the comparing and masking routine. The simulation was set up to test three sets of product variations, as shown in each column, to obtain the masking strings for each assembly device. The first comparison had no variations between the product strings (same product for all three). The result shows that all the parts to be placed are assigned to be done by the Cartesian, leaving the KUKA robot unused. This shows that the routine identifies matching features with success. The second comparison had no similarities between the product strings (three completely different products tested). This result shows that all the parts to be placed are assigned to the KUKA robot instead of the Cartesian. This furthermore shows that the routine accomplished to recognize the fluctuations between the product variations. The last comparison was done using typical product variations. The result shows that the workload is equally distributed between each assembly device. By implementing this routine, it becomes evident that the production handler is able to perform comparing and masking functions.

Table 5.1 Simulation results for masking handler

Strings values	Compare 1	Compare 2	Compare 3
Product 1	001100110011	011100101010	001111001011
Product 2	001100110011	111110000111	000111100010
Product 3	001100110011	100001011101	000100000111
Cartesian Mask	111111111111	000000000000	110100010010
KUKA Mask	000000000000	111111111111	001011101101

5.2.1.2 Priority Handler

To verify the priority handler, a simulation is set up using a virtual PLC and a virtual HMI. To expedite the simulation, the virtual assembly devices are included in simulation, and the simulation speed is increased to maximum. By doing this, device movements and operations cannot be observed clearly, but the order in which the products were built can be obtained faster. Furthermore, the simulation is set up by placing three batch orders in such a way so as to test different scenarios, as shown in Table 5.2. After ordering is completed, the system is instructed to start production and the simulation is underway.

Table 5.2 Setup for priority routine simulation

Scan list	Order 1		Order 2		Order 3	
	Priority	Quantity	Priority	Quantity	Priority	Quantity
Product A	1	2	3	2	2	2
Product B	1	2	2	2	1	2
Product C	1	2	1	2	2	2

After the simulation is completed, the results are obtained and tabulated as in Figure 5.4. The results show the sequence in which the products were built in each respective batch order as determined by their product priorities. In general, products with higher priority are built first. This is confirmed by results two and three. In addition, if the priorities are made equal, the products are built according to their order in the product scan list (A to C). If a product is scanned first, it is built first. This is corroborated by results one and three. Based on these results, it is clear that the production handler is able to prioritize the building order of products by utilizing this routine.

	1st	2nd	3rd	4th	5th	6th
Build Result 1	A	A	B	B	C	C
Build Result 2	C	C	B	B	A	A
Build Result 3	B	B	A	A	C	C

Figure 5.4 Sequence in which products were built

5.2.2 Testing System Capabilities with a Process Plan

5.2.2.1 Introduction

The first simulation test is done using the virtual KUKA robot and a process plan (refer to Figure 5.5). This method of verification simulates the behaviour of an assembly cell without connecting control logic to the cell. The intention of the test is to verify the operations of the KUKA assembly cell, to prove that the system can build more than one product on the same assembly line, shift workload to neighbouring conveyors, and evaluate the method of verification. To clarify, by using this method of verification, a process can be tested for collisions, and erroneous device operations can be rectified without damaging the real equipment.

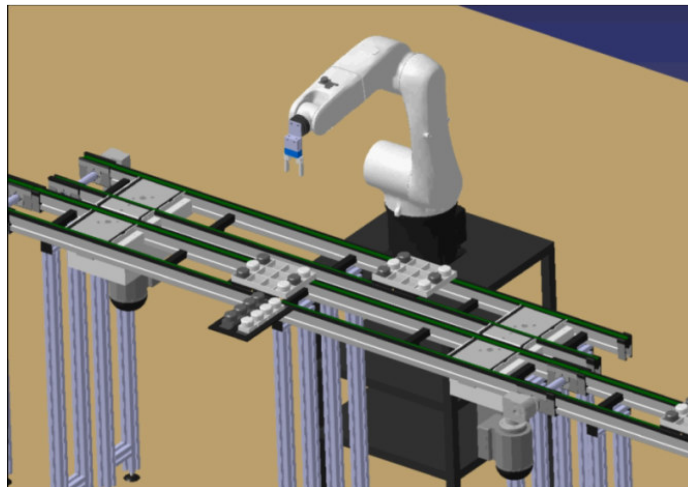


Figure 5.5 KUKA robot in virtual environment

5.2.2.2 Setup

To set up the process simulation, the environment is built up using the methods described in Chapter 3. Firstly, the operations the assembly cell must execute and the sequence in which the operations must occur have to be identified. Afterwards, the KUKA robot and the surrounding auxiliary devices in the cell are taught the various tasks needed to perform the identified operations, and then inserted into the process simulation as activities. These activities are shown in Figure 5.6. Lastly, these activities are linked together in the sequence that the activities should occur by using a PERT chart (shown in Figure 5.7). At this point, the process simulation is set up and is ready to be executed.

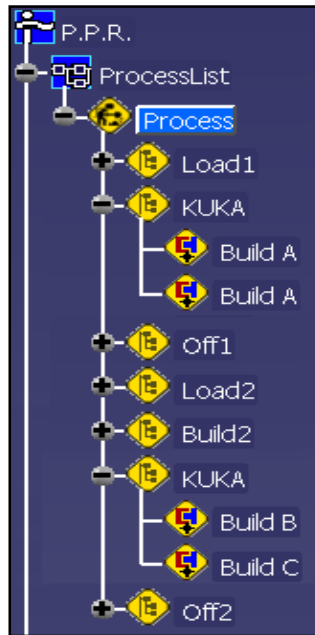


Figure 5.6 Processes present in product tree

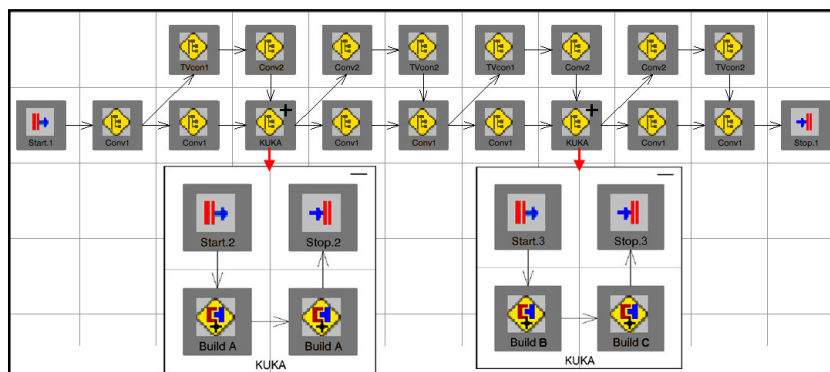


Figure 5.7 PERT chart implemented in process testing

5.2.2.3 Results

The process simulation is commenced and the result is shown in a support video in Appendix B.

The sequence starts with a conveyor activity that loads pallets (from the Cartesian robot assembly cell) onto both conveyor lines. After the pallets are in place, the KUKA robot is signalled to start assembling. For the first assembly process, the KUKA must build the same product on each of the conveyor lines

and executes it. This process demonstrates that the system is able to share workload among conveyor lines by building the same product on each conveyor line. Afterwards, the KUKA acknowledges that it finished assembly and the next conveyor activity is triggered. The conveyor now transports the finished products towards the gantry crane, and simultaneously loads new pallets for the next assembly process. When the new pallets are in place, the second KUKA process is triggered. For the second process, the KUKA must build two different products on each conveyor line. This firstly demonstrates that the system is capable of building more than one product, and secondly has the capability to build different products at the same time. After assembly is complete, the KUKA acknowledges and the last conveyor activity is triggered to offload the newly-built products.

5.2.2.4 Summary

To summarize, the test has shown the capability of the system to build multiple products on the same line and sharing the workload among conveyors. In addition to system capabilities, the operation of the KUKA robot was verified by using a process plan. Furthermore, by using a process plan as verification method, the system activities are confirmed with ease and can be altered in case of unpredictable operations. In contrast, the only drawback to using this method is that no control logic is involved in the simulation. This means that device operations can be tested and overall system capabilities be verified, but not the control logic which operates the system.

5.2.3 Verifying Control with a Virtual Programmable Logic Controller

5.2.3.1 Introduction

The second assembly cell verification test is done using the virtual gantry crane and a virtual PLC (refer to Figure 5.8). This method of testing is referred to as a “software in the loop” simulation. Here the cell verification is focused on how control logic effects the device operations. The intention of the test is to verify the operations of the gantry crane, to demonstrate how the system handles erroneous products, and evaluate this control verification method. Thus the work cell operations and movements can be verified using a virtual PLC, without needing to purchase a physical system PLC.

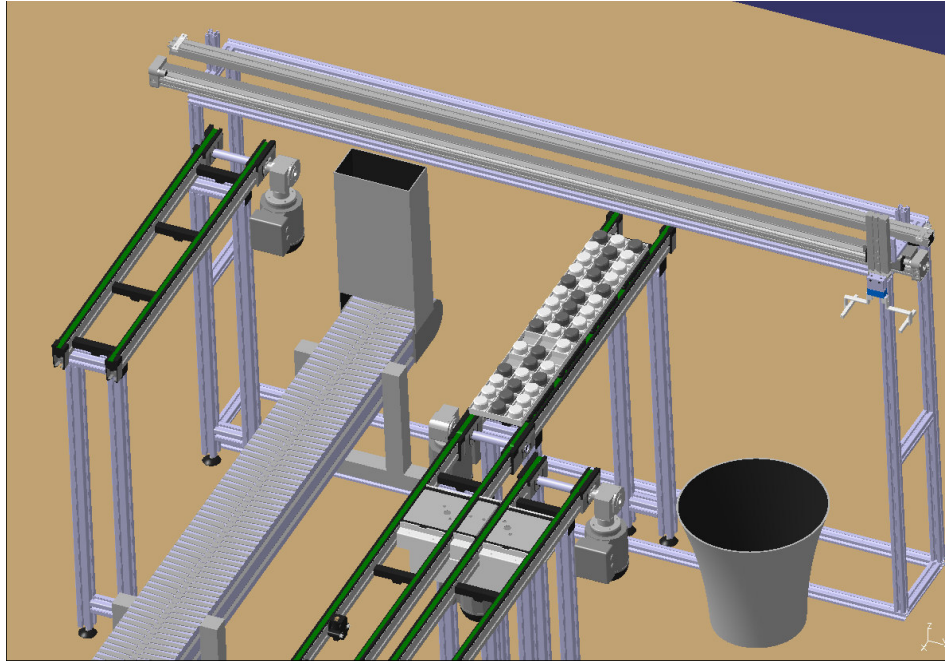


Figure 5.8 Gantry crane in virtual environment

5.2.3.2 Setup

The execution environment is set up by firstly preparing the geometry, then developing the control logic and afterwards mapping the signals.

Referring to Figure 5.8, the environment is prepared by placing four products on the outgoing conveyor, where one of these products is intentionally made faulty. This will simulate how the system handles erroneous products. Furthermore, the geometry is setup by placing a resource sensor at the end of the conveyor, to signal the virtual PLC that a product is in place. In addition, this resource sensor is tailored to return an integer value, to identify products like an RFID. In this way, the virtual PLC can discriminate between the detected products.

Next, the virtual PLC code to control the gantry operations is developed. This is done by using SFC and can be seen in Figure 5.9. For clarification, the gantry has internal control logic to control device operations and is interfaced with the supervisory virtual PLC to delegate the gantry operations. To conclude the setup, the devices and the virtual control logic are mapped to inform the execution environment how these components are interconnected. This is shown by Figure 5.10. At this point, the environment setup is prepared and ready to be executed.

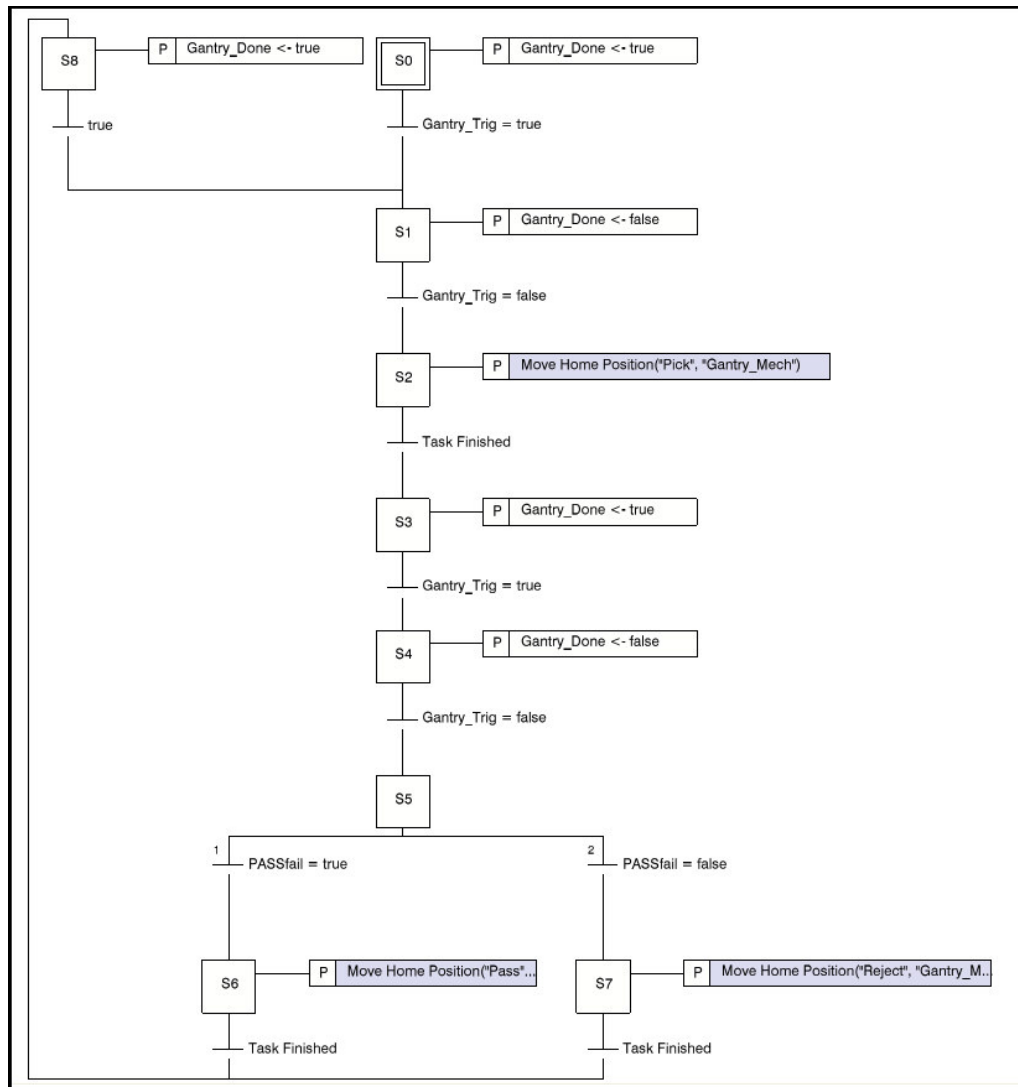


Figure 5.9 SFC code implemented in virtual PLC

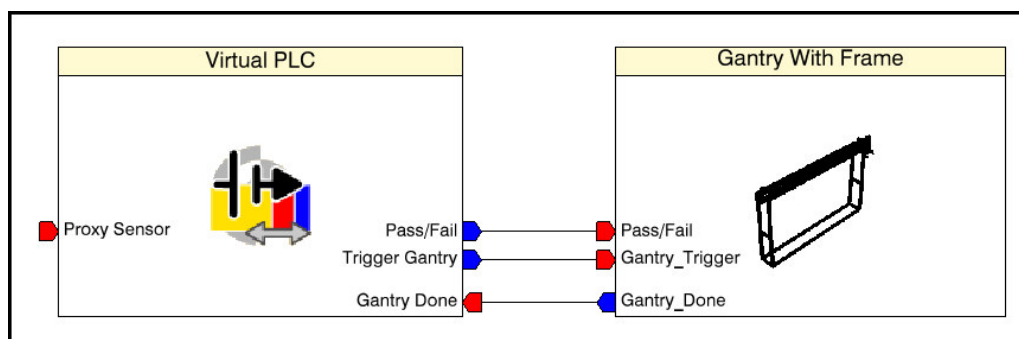


Figure 5.10 Interconnections between virtual PLC and gantry

5.2.3.3 Results

The execution environment is initiated and the result is shown by the video in Appendix C.

As a completed product reach the end of the conveyor, the virtual PLC detects its presence at the resource sensor and obtains an integer value (RFID value). Due to the fact that the visual inspection equipment for quality control cannot be connected to DELMIA for testing, simulation signals must be induced manually to inform the virtual PLC of faulty products. With this in mind, the first two products pass the quality check and the corresponding simulation signal to inform the PLC is provided. Afterwards, the PLC presents the gantry with a pass signal and instructs it to start operation. The gantry responds by picking up the product, moving to the pass position and placing the product on the dispatching conveyor. Afterwards, the gantry returns back to the picking position and acknowledges completion to the PLC. This procedure is repeated for the second product detected by the virtual PLC. For the third product, a simulation signal to inform the PLC that the current product is faulty is induced. Afterwards, the PLC provides a fail signal and instructs the gantry to execute the operation. The gantry reacts by picking up the faulty product, moving towards the reject bin and discarding the faulty product. Afterwards, the gantry returns to the picking position and acknowledges completion to the PLC. For the last product, the product is passed and the same procedure is followed as for the first two products, providing evidence that the system can distinguish between how to deal with good and faulty products.

5.2.3.4 Summary

To summarize, the test confirmed that the system can handle both good and faulty products, and verifies how the gantry operates in response to the virtual control logic. This method of verification provides an excellent way to validate the internal logic of a device and confirms that the control logic applied to the device provides the accurate signalling to operate the device. A drawback to using this method is that the actual PLC code to be implemented cannot be verified using this method. The reason for this is the coding language (ladder or SFC) used in DELMIA is not compatible with or similar to the equivalent used in the real system

PLC. Thus, only the functionality and methods of the control logic can be tested and not the actual code to be implemented.

5.2.4 Validating Logic with System Programmable Logic Controller

5.2.4.1 Introduction

The last assembly cell verification test is done using the virtual Cartesian robot and the real system PLC (refer to Figure 5.11). This is also referred to as a “hardware in the loop” simulation. The intention of the test is to verify the operations of the Cartesian assembly cell, validate function of the device handler, verify the actual control logic implemented on the PLC and evaluate the method of verification. Basically, this is a real-time system verification test, with emphasis on evaluating the actual controller code.

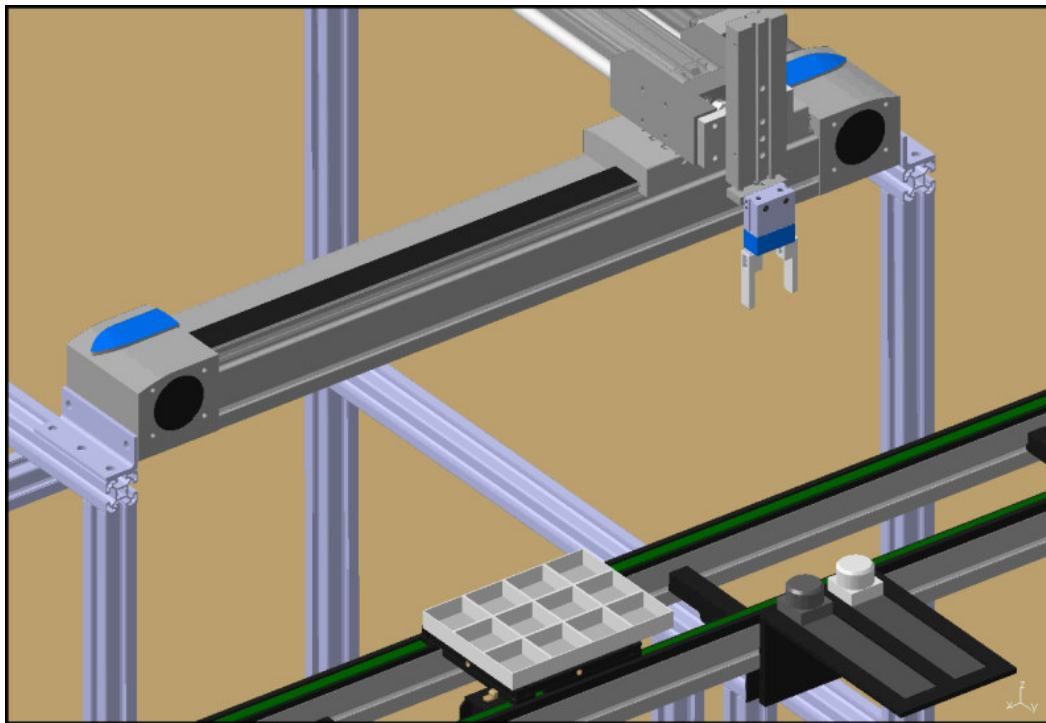


Figure 5.11 Cartesian robot in virtual environment

5.2.4.2 Setup

To set up the test environment, some functions and subroutines are disabled in the main PLC so that only the Cartesian robot and its auxiliary devices will be

able to operate. Next, the environment is set up in a similar manner to section 5.2.3.2, but in contrast an external (real) PLC is used instead of a virtual PLC. Furthermore, by using the methods from section 3.4.4 (setup an OPC server) and section 4.2.4 (setup of external PLC), the OPC tags are created, the path (IP address) specified and the connection (signal quality) between the OPC server and the environment is checked. Afterwards, the external PLC and the subcomponents of the Cartesian are interconnected as shown in Figure 5.12. To clarify, the control logic to operate the Cartesian is located inside the main controller; thus, the PLC must address the individual components separately. As a result, the setup of the execution environment is concluded and the simulation can commence.

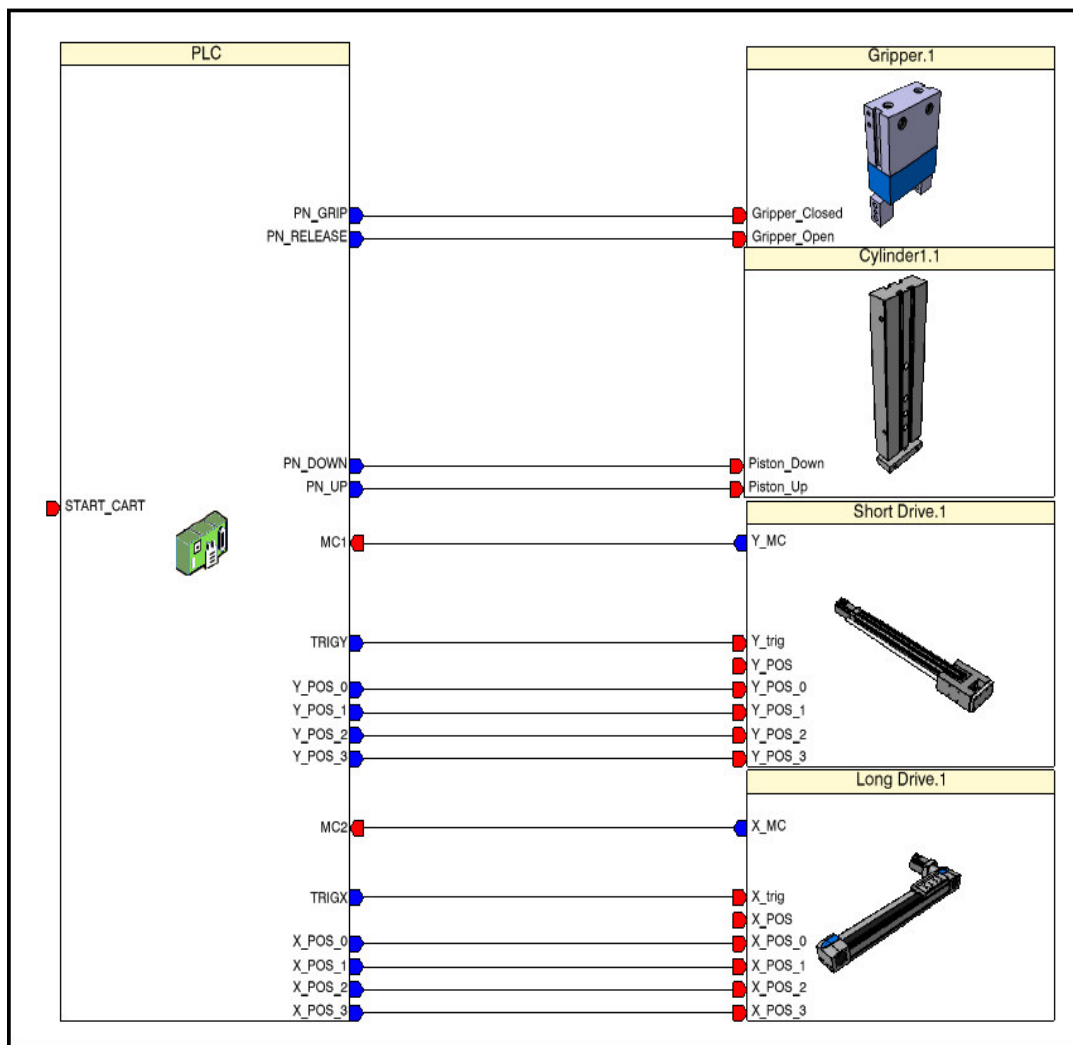


Figure 5.12 Interconnections between external PLC and Cartesian

5.2.4.3 Results

Since the real PLC is connected to the virtual environment, the virtual parts cannot be picked, placed or moved around in the environment. Due to this, only the movements and operations of the device can be shown and not the actual building. In addition, to start the emulation reliably, the execution environment is first initiated, and then instructions are sent to the PLC. The resulting process is shown in Appendix D. Afterwards, the PLC is provided with a product string to build the matching features between multiple products as determined by the method used in section 4.6.3.5. The resultant of this product to be built is shown in Figure 5.13.

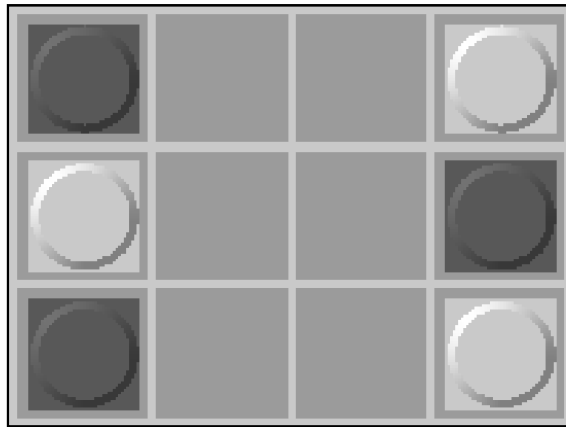


Figure 5.13 Matching features to be built by Cartesian

After the build data are provided to the PLC, the process is started by triggering the “START_CART” OPC tag present in the IO monitoring window (see Figure 5.14). The PLC responds by ensuring that the cylinder piston is up and the gripper is open (released). Afterwards, the PLC presents the Cartesian with X and Y coordinates (X_POS and Y_POS values) to direct the linear drives to the corresponding pick position and triggers the motion. Note that the motion complete outputs (MC1 and MC2) from the linear drives are false while the drives are in motion. After the drives are in position, the Cartesian responds with motion complete. Next, the PLC runs a pickup routine and signals the cylinder and gripper to perform the sequence. After the pickup sequence is completed, the PLC provides the Cartesian with placing coordinates and triggers the drives. Afterwards, motion complete is detected and the PLC runs a placing routine and

signals the cylinder and gripper to execute the sequence. This process continues until the product shown in Figure 5.13 is completely built and afterwards, the PLC remains idle until instructed to build the next product.

Name	Type	Status	Value
PLC			
MC1	Bool	—	true
MC2	Bool	—	true
PN_DOWN	Bool	■	false
PN_GRIP	Bool	■	false
PN_RELEASE	Bool	■	true
PN_UP	Bool	■	true
START_CART	Bool	—	true
TRIGX	Bool	■	false
TRIGY	Bool	■	false
X_POS_0	Bool	■	false
X_POS_1	Bool	■	true
X_POS_2	Bool	■	false
X_POS_3	Bool	■	false
Y_POS_0	Bool	■	true
Y_POS_1	Bool	■	false
Y_POS_2	Bool	■	true
Y_POS_3	Bool	■	false

Figure 5.14 IO monitoring window

5.2.4.4 Summary

To outline, the operations of the Cartesian as well as the actual PLC code are successfully verified in real-time. This method of verification provides a means to verify the internal logic of a device, the movements and operations, as well as the actual PLC code without the risk of damaging the real system equipment. Although movements of parts cannot be animated, virtual commissioning is successfully obtained and how the real system will operate is predicted. However, care should be taken to ensure that simulation speed corresponds to the processing speed of the PLC; otherwise, the virtual work cell performs unrealistic and erroneous device movements. To conclude, if a PLC of any brand can be browsed using an OPC server, the control logic and device operations can be verified through virtual commissioning.

5.2.5 Analysis - Compare the Different Cell Verification Methods

Based on the comparison done in Table 5.3, each method shows advantages and disadvantages. If emphasis is placed on early system layout, space reservation and process prediction, then method 1 is the suggested choice. On the contrary, if emphasis is placed on verifying control and device logic, along with cell operations, then methods 2 and 3 are highly recommended. To choose the supreme method to use depends mostly on what is intended to be achieved. Fortunately, DELMIA offers manufacturing companies a complete solution to test their designs well ahead of time, thus giving them the advantage over their opposition.

Table 5.3 Comparison between cell verification methods

	Method 1 (section 5.2.2) Process	Method 2 (section 5.2.3) Virtual Controller	Method 3 (section 5.2.4) Real Controller
Verify Device Movements	Yes	Yes	Yes
Verify Pick and Place Operations	Yes	Yes, but limited operation	No
Simulate Conveyor Action	Yes	Yes, but extra configuration setup is required	No
Collision Analysis	Yes	Yes	Yes
Verify Control Logic	No	Yes, but differ from actual control logic	Yes
Verify Device Logic	No	Yes	Yes

5.3 Test Done in Multi Agent System Mode

5.3.1 Test Setup

To test the system with the MAS in control, control is handed over to the MAS and the all the processing done by the PLC is inhibited. Figure 5.15 shows all the functions that are available to the MAS when this state is active. In addition to all the functions available to the MAS, the PLC switches to slave mode and runs the subroutine shown in Figure 5.16. By recalling that the MAS cannot ensure accurate and stable delay times (section 4.6.4), this routine performs delay tasks on behalf of the MAS. This results in the PLC executing all the tasks that the MAS instructs.

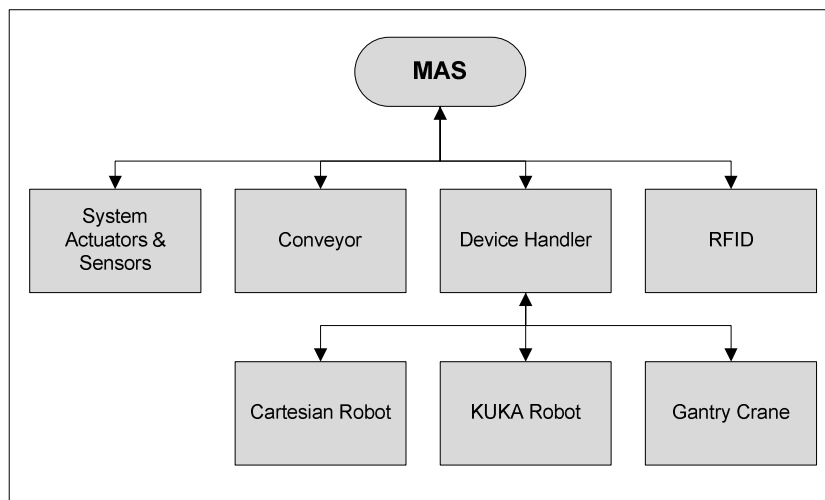


Figure 5.15 Active processes in MAS mode

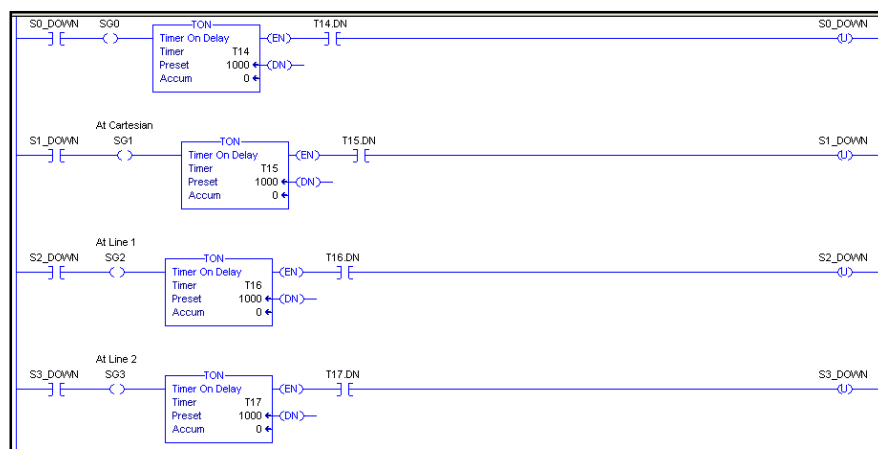


Figure 5.16 PLC in slave mode

5.3.2 Result

To reveal system operation with the MAS in control, an operator instructs the MAS to start production and a simple process is demonstrated.

The MAS starts the process by switching on the various conveyors and loading a pallet at the first (Cartesian) assembly cell. The MAS presents the device handler in the PLC with a product string and triggers the operation. The Cartesian executes the operations and the device handler acknowledges that the assembly operation is done. At this point, the MAS communicates to the vision system to execute a quality check. The product passes inspection and the MAS performs the necessary conveyor actions to let the pallet travel to the second (KUKA) assembly cell. Similarly to the Cartesian, the MAS provides the device handler with the corresponding product string, which the KUKA must build, and triggers the operation. The KUKA responds by building the second part of the product. Afterwards, the device handler informs the MAS that assembly is completed. The MAS once more enquires a quality check and the product passes inspection. The finished product is then conveyed towards the gantry crane. Unfortunately, the physical assembly of the gantry crane is not yet complete, but the movements and operations can be demonstrated without picking and placing the actual pallet. At this point, the MAS relays a pass signal to the device handler and the gantry is instructed to dispatch the product to the packaging conveyor. This reveals that the system is able to function flawlessly with the MAS in control.

5.4 Conclusion

This chapter revealed the testing done to verify the capabilities of the system and were divided into two parts, PLC and MAS in control of the system. In addition, a scrutiny of the methods of verification was done and the results obtained from it were presented. Also included in the chapter was a brief description of the system layout and where the wiring and network cables are located.

The first part of testing entailed validating the system with the PLC in control, and was performed by using DELMIA as the simulation environment. The tests validated the PLC process planner and its subcomponents, the device handler and each of the assembly devices present in the system. Furthermore, the tests

also examined the capabilities of the DELMIA verification methods and how virtual commissioning contributes towards system verification. These methods included system verification by a process plan, virtual logic and actual system control logic. The second part of testing was performed using the physical system with the MAS in control to verify overall system operation. This revealed that the MAS can execute a production process by accessing and controlling the IOs and subfunctions of the PLC; showed how the PLC operates in slave mode and responds to instructions from MAS.

To conclude, the results from simulations and testing revealed that the project objectives are achievable and that it is possible to build a system with these requirements. In addition, virtual commissioning can be utilized to expedite the validation of such a system. Furthermore, the results also showed that both the PLC and MAS control options are feasible, and that the utilization of either will depend solely on consumer preference. Finally, the comparative analysis which was done to inspect the capabilities of the DELMIA verification methods, revealed the most suitable methods for future use and it was found that virtual commissioning with real PLC control provides the most complete and accurate method of system verification for this project.

Chapter 6 Contributions and Conclusion

6.1 Introduction

This chapter briefly recapitulates the project, scrutinizes the research goals and objectives, lists the contributions made, and reveals future work to be done.

6.2 Summary

Chapter 1 introduced the project and placed it in perspective. It stated the problem, contained the hypothesis, research methodology and listed the objectives of the project.

In Chapter 2 a literature study was conducted to gain preliminary knowledge on the various aspects of a RAS. These aspects included characteristics of a RAS, design and flexibility issues, typically considered hardware and software components, and the use of PLM software in industry.

Afterwards, Chapter 3 introduced DELMIA as the test environment. It revealed how the infrastructure works, how to obtain geometry and specifying its behaviour, and how to accomplish control element verification through virtual commissioning.

Later, chapter 4 documented on which basis the system components were chosen and how they were implemented in the system. The implementation in Chapter 4 revealed the necessity to develop various methods and software modules to control and operate the system successfully. This included methods on how an operator interacts with the system to place orders, how the system translates these orders into a process plan, and how the process is executed afterwards. In addition, OPC and KEPserver were introduced along with how these are set up and used as a communication platform. Another aspect was the two overhead control instances and how the control was handed over between these instances by the system.

Ultimately, Chapter 5 presented the tests which were done to verify the various objectives of the project. It showed how the tests were prepared, the simulations executed and the results which were obtained afterwards to verify critical system

components. In addition to simulations, the physical system was also tested with the MAS in control. This test showed that the MAS was able to control a production process by using OPC as the communication platform and verified the MAS as a control instance.

6.3 Research Goals and Objectives

The main goals of the project were to develop a RAS with enhanced control capabilities which can switch control between the MAS and system PLC, build more than one product on the same assembly line, distribute the workload among system components, and handle erroneous products in the system. In addition, this had to be achieved by creating and utilizing a virtual simulation environment to verify the operation and control of the system.

This was met through firstly building a virtual version of the system by using the methods discussed in Chapter 3. In addition, the subcomponents used in the virtual system were given the equivalent functionality and behaviour as their physical counterparts as chosen and specified in Chapter 4. Afterwards, the control logic for the system was developed as shown in Chapter 4 and combined with the virtual system to perform the verification testing done in Chapter 5. Also in Chapter 5, the physical system was used to verify how the system performs with the MAS in control as well as other subfunctions incorporated in the system.

6.4 Contributions

The project delivered the following contributions:

6.4.1 Enhanced Control (State Machine)

A flexible and reconfigurable state machine was developed as part of the main PLC code to control the system. It consists of various operational states, where the state machine chooses between MAS and PLC control by transferring to one of these operational states. In addition to the state machine, software functions are developed to be modular so that they can be easily added or removed from the system software, or be enabled or disabled in a particular operational state. Moreover, when an operational state is selected, particular functions are enabled and executed, and the redundant and unused features disabled to provide the

selected state with the required functionalities. This enhances the control capabilities and options of the system PLC.

6.4.2 Production Planner

A production planner was developed to aid the PLC to handle production and assembly processes when in the “internal process controller state”. The production planner consists of a priority routine to determine the order in which products are built, a masking routine to determine the workload at each assembly cell, and an RFID handler to track products in the system. Essentially the function of the production planner is to decide “what” is built “where,” and “in which order”.

6.4.3 Device Handler (Build Algorithm)

A device handler was implemented to operate the various assembly devices in the system. A flexible build algorithm to determine the operations and instruct the respective assembly devices was developed to provide functionality to the device handler. The device handler seizes control over device operations and executes recipes provided from both PLC and MAS, respectively, when either one has control.

6.4.4 Simulation Model and Virtual Commissioning

A virtual simulation model of the entire system was created using CATIA and DELMIA. All the CAD models for the subcomponents used to build up the virtual system are designed at a 1:1 ratio compared to the real system. Furthermore, these subcomponents were assembled into the respective system assembly devices by creating mechanical joints, moving constraints and the internal logic to specify behaviour for each. Afterwards, the devices were arranged into their proposed positions in the virtual version of the laboratory. This forms the basis of all current and future simulation testing done on the system.

In addition to the virtual infrastructure, a virtual version (code was modified to run in DELMIA) of the overhead control software was also developed, which will enable future students to test new implementations or modifications without changing the physical system. Furthermore, an OPC server was set up and communication between it and the DELMIA environment was established,

enabling connections with the real system PLC. This ensures that the movements, operations, processes and control logic of the virtual system can be verified, thus establishing virtual commissioning for future use. In addition to virtual commissioning, the solution shown in section 3.3 was contributed to simulate the actions and operation of the system conveyor.

As a result, future students have access to a complete simulation platform, which enables them to introduce new projects into the system easily, modify current designs with ease and verify the changes made in both software and hardware using virtual commissioning.

6.4.5 Physical System Infrastructure

A complete physical version of the system has been built in the RGEMS research laboratory. This includes that all the assembly devices were assembled from their various subcomponents; the devices were placed and secured into their positions in the system; the respective wiring and communication connections were made with the controlling instances and their required EoA tools were mounted. Furthermore, the conveyor joining the assembly devices was assembled, and its driving motors, proximity sensors and actuators were also mounted and connected. In addition to the physical components, an Ethernet and DeviceNet network has been installed to communicate between the control instances and the various assembly devices. Furthermore, precautionary safety devices were implemented, as well as an enclosing physical structure that was built to ensure the safety of laboratory personnel. As a result, current and future RGEMS students will benefit by having access to a full system, having a range of assembly devices, which enable them to complete their projects faster, more easily and safely.

6.5 Future Work

6.5.1 Simulation Environment

The current simulation environment functions satisfactorily for the project, but different methods and possibly evaluation of other software packages is worth

looking into to find the best, up-to-date practices and PLM software available to the manufacturing industry.

6.5.2 Error Handling

The system currently only tests if certain software modules are available and if hardware devices are active and responds to instructions. However, methods to resolve system errors like hardware or software component failure must still be developed and implemented. Some recommendations concerning this include that the system might automatically bypass a faulty component, reroute products to other assembly lines, and shift the workload of the faulty component to the remaining operational components to perform. Furthermore, software might be modified to test for errors and solve the errors without the need for operator intervention.

6.5.3 Tool Reconfigurability

Considerations to further improve the reconfigurability of the system include the implementation of either multi tools or tool changers. A multi tool, for instance, consists of various tools (gripper, sucker and a glue gun) mounted on the flange of a robot at different angles and changes the posture or approach of the robot to use it. In contrast, a tool changer consists of a tool gripper, which picks up a tool needed for an application from a tool tray and substitutes it with another when the current operation is complete. Adding tool flexibility to the system devices will increase the assembly capability and enhance the overall reconfigurability of the system.

6.5.4 Rework Bay

Presently if the system encounters an erroneous product it discards it without considering why it is defective. Plans to add an addition shunt conveyor system with incorporated machine vision and assembly robots are considered for future endeavours. The purpose of this shunt conveyor system will be to reconsider the condition of the product and determine the immensity of the errors on it. It would be considered to rather rework the product by fixing the minor errors instead of discarding it without consideration.

6.6 Conclusion

Most of SA manufacturing exports are aimed at niche markets, the majority of which involve high varieties of products in small volumes. Furthermore, SA has a large availability of manual labour to accommodate for the manufacturing of this diversity in products, but the quality of these products cannot be ensured by utilizing manual labour alone. In addition, cases where manufacturers do utilize automation and techniques, a tendency towards dedicated assembly lines are pursued and this restricts the flexibility which the products in demand require. This then clearly indicates why SA should pursue utilizing RAS to provide a flexible platform that is needed to accommodate the required diversity in product fabrication. This will result in SA manufacturing industries having reconfigurable systems which can handle rapid introduction of new products and assemble multiple product variations at variable volumes with no concern of short product lifecycles. It then becomes evident that the possibilities are endless; instead of having numerous discrete dedicated systems to perform various devoted tasks and processes, these same tasks can be achieved by one system simply by reconfiguring it. In addition, by utilizing PLM software to design these systems and verifying it by obtaining virtual commissioning, the development time can be decreased, modification to an existing system can be made with ease, and the final system can be quicker operationally. This will give the SA manufacturing industry a further advantage, which makes PLM software a must-have tool in RAS development and maintenance. However, the long-term benefits that the utilization of RAS can provide to SA include that the production and quality of products will increase and the product cost will decrease, which results in an increase of revenue. This will restore the trust of foreign companies in SA, which will draw investors and increase economical security in SA. Additionally, the skills requirements to build, maintain, and operate these systems will create job opportunities for professionals which will further develop SA and its communities. In short, the implementation of RAS will benefit SA tremendously and enable it to compete for global markets. This raises the ultimate question. How can the SA manufacturing industry not be more competitive, more productive, limit expenses and show a profit if they utilize RAS?

References

- [1] Y. Koren, "Reconfigurable Manufacturing Systems," *COMA annals*, vol. 1, pp. 69-79, 2004.
- [2] N.F. Edmonson and A.H. Redford, "Generic flexible assembly system design," *Assembly automation*, vol. 22, no. 2, pp. 139-152, 2002.
- [3] T. Raj, R. Shankar and M. Suhaid, "A review of some of the issues and identification of some barriers in the implementation of FMS," *International Journal of Flexible Manufacturing Systems*, vol. 19, no. 1, pp. 1-40, 2007.
- [4] 3DS - Dassault Systemes, Available Online:
<http://www.3ds.com/products/delmia/products/all-delmia-products>, last accessed in May 2011.
- [5] CDC - CNC Design Consultants, Available Online:
<http://www.cdcza.co.za/category/software/delmia>, last accessed in May 2011.
- [6] Rockwell Automation, Available Online:
<http://ab.rockwellautomation.com/Networks-and-Communications/DeviceNet-Network>, last accessed in July 2011.
- [7] H.P. Wiendahl, "Changeable Manufacturing - Classification, Design And Operation," *Annals of CIRP*, vol. 56, pp. 783 - 809, 2007.
- [8] R.M. Setchi and N. Lagos, "Reconfigurability and Reconfigurable Manufacturing Systems - State of the art Review," in *2nd IEEE International Conference of Industrial Informatics: Collaborative automation - one key for industrial environments*, Berlin, 2004.
- [9] M. N. Rooker, C. Sunder, A. Zoitl, O. Hummer and G. Ebenhofer, "Zero Downtime Reconfiguration of Distributed Automation Systems," in *Proceeding of the 3rd International Conference on industrial applications of Holonic and Multi-agent Systems*, Regensburg, 2007.
- [10] Y. Koren, U. Heisel, F. Jovane, G. Pritschow, T. Moriwaki, A.G. Ulsoy and H. Brussel, "Reconfigurable Manufacturing Systems," *CIRP - Manufacturing Technology*, vol. 48, no. 2, pp. 527-540, 1999.

- [11] H. ElMaraghy, "Flexible and Reconfigurable Manufacturing Systems Paradigms," *International Journal of Flexible Manufacturing Systems*, vol. 17, no. 2, pp. 1-13, 2006.
- [12] R. Katz, "Design principles of reconfigurable machines," *The international journal of Advanced Manufacturing Technology*, vol. 34, no. 5, pp. 430-439, 2007.
- [13] Britannica, Available Online: <http://global.britannica.com/EBchecked/topic/44912/automation/24847/The-robot-manipulator>, last accessed in March 2011.
- [14] RobotWorx, Available Online: <http://www.robots.com/glossary>, last accessed in March 2011.
- [15] RobotWorx – Welding Division, Available Online: <http://www.welding-robots.com/faq.php?question=robot+work+envelope>, last accessed in March 2011.
- [16] RobotWorx, Available Online: <http://www.robots.com/faq/show/what-are-the-main-types-of-robots>, last accessed in April 2011.
- [17] KUKA-Robotics, Available Online: www.KUKA.com, last accessed in March 2011.
- [18] RobotWorx, Available Online: <http://www.used-robots.com/articles.php?tag=1557>, last accessed in April 2011.
- [19] ROBOTIQ, Available Online: <http://blog.robotiq.com/bid/33127/How-To-Choose-The-Right-Robotic-Gripper-For-Your-Application>, last accessed in April 2011.
- [20] Crafting Material Interfaces, Available Online: <http://material.media.mit.edu/?p=1451>, last accessed in April 2011.
- [21] Inovative Conveyor Concepts, Available Online: <http://www.icconveyors.com/>, last accessed in April 2011.
- [22] Emiplastics, Available Online: <http://www.emiplastics.com/Conveyor/>, last accessed in April 2011.

- [23] Daifuku Webb – Jervis B. Webb Company, Available Online: <http://www.jervisbwebb.com/Categories/AGVs.aspx?cid=3>, last accessed in May 2011.
- [24] Articlebase, Available Online: <http://www.articlesbase.com/industrial-articles/different-types-of-actuators-2818876.html>, last accessed in May 2011.
- [25] Trade Korea, Available Online: [http://www.tradekorea.com/product-detail/P00283023/DNC air cylinder Festo standard pneumatic cylinder.html](http://www.tradekorea.com/product-detail/P00283023/DNC%20air%20cylinder%20Festo%20standard%20pneumatic%20cylinder.html), last accessed in May 2011.
- [26] Sharif University of Technology, Available Online: http://ee.sharif.edu/~industrialcontrol/Summary_Automation_Sensors_tutorial.pdf, last accessed in March 2011.
- [27] Direct Industry, Available Online: <http://www.directindustry.com/prod/crouzet/inductive-proximity-sensors-7457-483212.html>, last accessed in March 2011.
- [28] Rockwell Automation, Available Online: <http://www.rockwellautomation.com/products-technologies/control-systems/overview.page>, last accessed in March 2011.
- [29] Fieldbus Inc, Available Online: http://www.fieldbusinc.com/downloads/fieldbus_comparison.pdf, last accessed in June 2011.
- [30] RT Automation, Available Online: <http://www.rtaautomation.com/devicenet/>, last accessed in June 2011.
- [31] OnBarcode.com, Available Online: http://www.onbarcode.com/code_128/, last accessed in June 2011.
- [32] OnBarcode.com, Available Online: http://www.onbarcode.com/data_matrix/, last accessed in June 2011.
- [33] OnBarcode.com, Available Online: http://www.onbarcode.com/data_matrix/data_matrix_size_setting_in_csharp.html, last accessed in June 2011.
- [34] Webopedia - RFID ref, Available Online: <http://www.webopedia.com/TERM/RFID.html>, last accessed in June 2011.

- [35] Machinevision.ca, Available Online: [http://www.machinevision.ca/files/Brochures/Introduction to Machine Vision.pdf](http://www.machinevision.ca/files/Brochures/Introduction%20to%20Machine%20Vision.pdf), last accessed in June 2011.
- [36] Allwords, Available Online: <http://www.allwords.com/word-manumation.html>, last accessed in June 2011.
- [37] Rover Ranch, Available Online: <http://prime.jsc.nasa.gov/ROV/types.html>, last accessed in June 2011.
- [38] ST Robotics - Why use a robot?, Available Online: <http://www.strobotics.com/whyrobot.htm>, last accessed in June 2011.
- [39] Machinery Safety 101, Available Online: <http://machinerysafety101.com/series/emergency-stop/#axzz2QcxlrZzg>, last accessed in July 2011.
- [40] Rockwell Automation, Available Online: <http://ab.rockwellautomation.com/Safety/Sensors-Switches>, last accessed in July 2011.
- [41] Occupational Safety and Health Administration, Available Online: http://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html, last accessed in July 2011.
- [42] Osh.net - Working Safely Around Industrial Robots, Available Online: http://www.osh.net/articles/archive/osh_basics_2002_may24.htm, last accessed in July 2011.
- [43] E. Mitka, N. Kyriakoulis, A. Gasteratos and S. G. Mouroutsos, "Safety certification requirements for domestic robots"(3 levels)," *Safety Science - ISSN 0925-7535, 10.1016/j.ssci.2012.05.009*, vol. 50, no. 9, pp. 1888-1897, 2012.
- [44] Loop Technology, Available Online: <http://www.looptechnology.com/robotic-robot-safety.asp>, last accessed in July 2011.
- [45] Robotics Online, Available Online: http://www.robotics.org/content-detail.cfm/Industrial-Robotics-Tech-Papers/Choice-of-Protective-Measures/content_id/154, last accessed in July 2011.
- [46] Antenen, Available Online: <http://www.antenen.com/htdocs/pre-eng-cells.html>, last accessed in July 2011.

- [47] Allen Bradley, Available Online:
<http://www.ab.com/en/epub/catalogs/3377539/5866177/3377569/6388297/383633/10197357/>, last accessed in June 2011.
- [48] Siemens - Technomatix, Available Online:
http://www.plm.automation.siemens.com/en_us/products/tecnomatix/index.shtml, last accessed in August 2011.
- [49] WinMOD, Available Online: <http://www.mewes-partner.de/en/>, last accessed in August 2011.
- [50] Visual Components, Available Online:
<http://www.visualcomponents.com/Products/3DAutomate>, last accessed in August 2011.
- [51] Virtual Commissioning, Available Online: <http://www.virtual-commissioning.com/products/>, last accessed in August 2011.
- [52] 3DS - Dassault Systemes, Available Online:
http://www.3ds.com/fileadmin/PRODUCTS/DELMIA/PDF/DM-12877-Robotics-Virtual-Commissioning-Datasheet_HR.pdf, last accessed in December 2011.
- [53] PMC Data - Driven Productivity Solutions, Available Online:
http://www.pmc corp.com/Portals/5/Downloads/b07_Process%20Simulate%20Virtual%20Commissioning.pdf, last accessed in August 2011.
- [54] 3DS - Dassault Systemes, Available Online: <http://www.3ds.com>, last accessed in March 2011.
- [55] CDC - CNC Design Consultants, Available Online:
<http://www.cdcza.co.za/category/software/catia>, last accessed in March 2011.
- [56] CATIA Online Documentation, Available Online:
<http://www.catiadesign.org/doc/catia/B205doc/English/online/DSDoc.htm>, last accessed in September 2011.
- [57] DELMIA Online Documentation V5R19, Available Online:
http://help.plmchina.cn/delmia/online/DELMIA_default.htm, last accessed in September 2011.

- [58] CATIA Online Documentation, Available Online:
http://catiadoc.free.fr/online/CATIAfr_C2/prtugCATIAfrs.htm, last accessed in October 2012.
- [59] DS Documentation Portal - CATIA, Available Online:
http://catiadoc.free.fr/online/CATIAfr_C2/asmugCATIAfrs.htm, last accessed in October 2012.
- [60] DELMIA Online Documentation V5R19, Available Online:
http://help.plmchina.cn/delmia/online/DELMIAfr_D2/dbqugDELMIAfrs.htm, last accessed in October 2012.
- [61] DELMIA Online Documentation V5R19, Available Online:
http://help.plmchina.cn/delmia/online/DELMIAfr_D2/wsugDELMIAfrs.htm, last accessed in November 2012.
- [62] DELMIA Online Documentation V5R19, Available Online:
http://help.plmchina.cn/delmia/online/DELMIAfr_D2/lb1ugDELMIAfrs.htm, last accessed in November 2012.
- [63] DELMIA Online Documentation V5R19, Available Online:
http://help.plmchina.cn/delmia/online/DELMIAfr_D2/dldugDELMIAfrs.htm, last accessed in November 2012.
- [64] DELMIA Online Documentation V5R19, Available Online:
http://help.plmchina.cn/delmia/online/DELMIAfr_D2/le1ugDELMIAfrs.htm, last accessed in November 2012.
- [65] DELMIA Online Documentation V5R19, Available Online:
http://help.plmchina.cn/delmia/online/DELMIAfr_D2/prpugDELMIAfrs.htm, last accessed in November 2012.
- [66] DELMIA Online Documentation V5R19, Available Online:
http://help.plmchina.cn/delmia/online/DELMIAfr_D2/apnugDELMIAfrs.htm, last accessed in November 2012.
- [67] DELMIA Online Documentation V5R19, Available Online:
http://help.plmchina.cn/delmia/online/DELMIAfr_D2/wsugDELMIAfrs.htm, last accessed in November 2012.

- [68] DELMIA Online Documentation V5R19, Available Online:
http://help.plmchina.cn/delmia/online/DELMIAfr_D2/dc1ugDELMIAfrs.htm,
last accessed in December 2012.
- [69] DELMIA Online Documentation V5R19, Available Online:
http://help.plmchina.cn/delmia/online/DELMIAfr_D2/hp1ugDELMIAfrs.htm,
last accessed in December 2012.
- [70] DELMIA Online Documentation V5R19, Available Online:
http://help.plmchina.cn/delmia/online/dldug_D2/dldugat0400.htm, last
accessed in December 2012.
- [71] OPC Foundation, Available Online:
[http://www.opcfoundation.org/default.aspx/01_about/01_what_is.asp?mid=ab
outopc](http://www.opcfoundation.org/default.aspx/01_about/01_what_is.asp?mid=aboutopc), last accessed in August 2011.
- [72] Kepware Technologies - OPC Servers / Communications for Automation,
Available Online: <http://www.kepware.com>, last accessed in September
2011.
- [73] Scribd - KEPserverEX V5 Help, Available Online:
<http://www.scribd.com/doc/63419485/KepserverEx-v5-Manual>, last
accessed in September 2011.
- [74] Bosch Rexroth, Available Online: [http://www13.boschrexroth-
us.com/catalogs/mit/ts1/TS1_Catalog_v3.pdf](http://www13.boschrexroth-us.com/catalogs/mit/ts1/TS1_Catalog_v3.pdf), last accessed in August
2011.
- [75] T. Zeifang, "The Optimization of a Reconfigurable Assembly System,"
*Master Dissertation, Faculty of Engineering and Information Technology,
Central University of Technology, Free State, 2010.*
- [76] Kiowa, Available Online: [http://www.kiowa.co.uk/products/HGP-16-A-B-
Festo-Parallel-gripper/P00044253/330](http://www.kiowa.co.uk/products/HGP-16-A-B-Festo-Parallel-gripper/P00044253/330), last accessed in August 2011.
- [77] FESTO - Support Portal, Available Online:
http://www.festo.com/net/tr_tr/SupportPortal/default.aspx?cat=1530, last
accessed in March 2011.
- [78] FESTO - Stepper motor EMMS-ST and motor controller CMMS-ST,
Available Online: http://www.festo.com/cms/nl-be_be/10488.htm, last
accessed in March 2011.

- [79] FESTO - FCT Manual, Available Online: http://www.festo-didactic.com/download.php?name=FCT_manual.pdf&c_id=1100&file=fct_manual.pdf, last accessed in March 2011.
- [80] KUKA-Robotics, Available Online: http://www.kuka-robotics.com/norway/en/products/industrial_robots/special/scara_robots/kr5_sixx_r850/, last accessed in July 2011.
- [81] Traceparts, Available Online: http://www.tracepartsonline.net/PartsDefs/Production/KUKA_PDF/10-03032009-072118/documents/KUKA_KR5sixx_Spezifikation_en.pdf, last accessed in July 2011.
- [82] KUKA System Software 5.5 Manual, Available Online: <http://sites.poli.usp.br/d/PMR2560/Manual%20KUKA.pdf>, last accessed in July 2011.
- [83] Scribd - KRC2 Expert Programming Manual, Available Online: <http://www.scribd.com/doc/86788986/KRC2-Expert-Programming-Manual>, last accessed in July 2011.
- [84] Rockwell Automation, Available Online: http://literature.rockwellautomation.com/idc/groups/literature/documents/sq/1756-sg001_en-p.pdf, last accessed in September 2011.
- [85] Allen Bradley, Available Online: <http://ab.rockwellautomation.com/Graphic-Terminals/2711P-PanelView-Plus-700>, last accessed in March 2012.
- [86] J. Janse van Rensburg, "Process Control and Configuration in a Reconfigurable Production System Using a Multi Agent Software System," *Master Dissertation, Faculty of Engineering and Information Technology, Central University of Technology, Free State, 2012.*

Appendices

The following appendices can be attained from the attached CD:

Appendix A: Conveyor

Appendix B: KUKA process simulation

Appendix C: Gantry using virtual PLC

Appendix D: Cartesian virtual commissioning using system PLC