TAMPERE UNIVERSITY OF TECHNOLOGY

KAN LI

3D MODELLING AND SIMULATION OF A PRODUCTION LINE WITH CIROS

Master's thesis

Examiner: Professor Jose Lastra

Dr. Andrei Lobov

Examiner and topic approved by the Faculty Council of the Faculty of Automation, Mechanical and Materials Engineering on7th December 2011.

PREFACE

It has been a long and wonderful journey to finalize this paper, which is also a major milestone of my study life. There is no accurate words could express my feeling at this moment. I have been dreaming to come to this day to finally thank the people who have been around me all this time.

I want to convey my biggest love to my parents, who have been my rock ever since the first day I decided to come to this white land to pursue my dream. You guys have been so supportive and understanding and I would not conquer so many challenges without your generous love. My dear friend Tian Song and Xu Li who made my life here in Finland into so much fun that I will never forget in my whole life. My best friends in China, Zhang Chenxi, Shao Wen, Xu Jingshu and Zhang Shu, you guys know that how much I love talking and sharing every little thing in my life with you and I cannot wait to see you guys soon.

I would also like to deliver my greatest appreciation to my supervisor Dr. Andrei Lobov and Professor Jose Lastra. I still have vivid memory of struggling in the work for many days and would not have a clear outlet if it is not for the help from Andrei. Also I will remember having the biggest trouble to locate my lovely supervisor, which has turned out to be an indescribable fun during my work. I would also like to thank my colleges in FAST-lab, Peymen, Prasad, Dazhuang, Juhani, Ahmed, who have made my school life so fruitful.

Three and half years in Tampere, Finland, everything I learn and everyone I know here have become a tremendous treasure. I will miss all of it for the rest of my life.

Li Kan

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY

Degree Programme in Machine Automation

Li, Kan: 3D modelling and simulation of a production line with CIROS

Master of Science Thesis, 69 pages, 9 Appendix pages

December 2011

Major subject: Factory automation

Examiner: Professor Jose Lastra, Dr. Andrei Lobov

Keywords: 3D simulation, Flexlink, FESTO CIROS Studio, Siemens STEP7

3D simulation technology has been adopted successfully in production industry for decades. It benefits the manufacturers by the possibility to answer ‘how would it be’ with vivid visual images, consuming much lower capital investment, resources and human power.

This thesis paper first investigates into the background research of simulation and modelling approaches employed within the industry. Then a pallet-based Flexlink production line in FAST-Lab, Tampere University of Technology, is taken as the simulated object for case study. 3D model is created under FESTO CIROS Studio software environment, using built-in mechanism offered by the program to realize full transportation system of the assembly line, both sensors and actuators. Logic control of the conveyor system is integrated with built-in virtual PLC and programmed in FBD and STL with Siemens STEP7.

The assessment results reveal the possibility of handling multiple pallets with multiple recipes simultaneously. Also the performance of FESTO CIROS Studio is evaluated as showing some limitations during research.

CONTENTS

1. Introduction 8

1.1. Scope 8

1.2. Outlines 8

2. Background research of simulation 10

2.1. Introduction to simulation 10

2.1.1. Pros and Cons 11

2.1.2. Simulation process 12

2.2. Manufacturing simulation 14

2.2.1. Benefits 14

2.2.2. Classification 14

2.3. Flexible Manufacturing System (FMS) 15

2.4. Multi-Agent Simulation (MAS) 17

2.5. Petri Net 19

2.5.1. Formal definition of Petri Net 19

2.5.2. Properties of Petri Net 20

2.5.3. PN Application 21

2.5.4. Advantages of PN in production simulation 24

2.6. CAD/CAM tools 24

2.6.1. FESTO CIROS Studio 25

2.6.2. QUEST 26

2.6.3. Taylor Enterprise Dynamics 27

2.6.4. Visual Components product family 28

2.6.5. Comparison 30

3. Case study presentation 32

3.1. Layout of assembly line 32

3.1.1. Start segment 33

3.1.2. Middle segment 34

3.1.3. End segment 36

3.2. Routing description 37

3.3. Mechanism application and I/O configuration 39

3.3.1. Actuators 39

3.3.2. Sensors 41

3.4. Design limitation and solutions 42

4. Simulation implementation 48

4.1. Implementation steps 48

4.1.1. Insert the PLC into the model 48

4.1.2. Link the I/Os of the PLC with the objects in the model 49

4.1.3. PLC programming 50

4.2. Design specifications 50

4.2.1. Timing for pallet stopping 51

4.2.2. Interactions between segments 51

4.2.3. Multiple pallets handling 52

4.2.4. Decision signals 53

4.3. User Interface and recipe loading 54

4.4. Pallet tracking 56

4.5. Process perfection 58

4.5.1. Process scan 58

4.5.2. Final scan 59

5. Assessment analysis 60

5.1. Simulation results 60

5.1.1. Single recipe execution 60

5.1.2. Multiple recipes execution ... 62

5.2. KPI analysis ... 64

6. Conclusion ... 66

References ... 68

Appendix 1 ... 70

Appendix 2 ... 72

ABBREVIATIONS

3D	Three-dimensional
BCL	Batch Control Language
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CNC	Computer Numerical Control
DAS	Dynamic Assembly System (Flexlink Product Concept)
DEC	Discrete-Event Control
DSS	Decision Support System
FBD	Function Block Diagram
FESTO CIROS	Festo corporation simulation software
FMS	Flexible Manufacturing System
IL	Instruction List
IPC	Industrial Personal Computer
IRL	Industrial Robot Language
KPI	Key Performance Indicator
LD	Ladder Diagram
MAS	Multi-Agent System
PLC	Programmable Logic Controller
PN	Petri Net
PNDEC	Petri Net based Discrete Event Controller
QUEST	QUeueing Event Simulation Tool (Delmia)
SCL	Simulation Control Language
SFC	Sequential Function Chart
SIPN	Signal Interpreted Petri Net
ST	Structure Text

List of Figures

Figure 1 Vehicle simulator....10
Figure 2 Process of simulation....13
Figure 3 Major components of FMS and their relationships [Colombo, 2010]....16
Figure 4 Agent-based Information Technology Fusion in Mechatronics [Colombo, 2010]....17
Figure 5 Example of agent interactions in manufacturing control [Colombo, 2010]....18
Figure 6 Collaborative production automation architecture [Colombo, 2010]....19
Figure 7 PN graphic example....20
Figure 8 Sample PN model for production events [Colombo, 2010]....22
Figure 9 Block diagram of PNDEC [Korotkin et al, 2010]....23
Figure 10 FESTO CIROS Studio screenshot....25
Figure 11 Taylor ED, 2D model with connected channels [Incontrol, 2011]....28
Figure 12 Taylor ED, 3D model of three counters [Incontrol, 2011]....28
Figure 13 3DRealize interface [Visual Components]....29
Figure 14 Flexlink production line....32
Figure 15 Top view of FESTO Studio model....33
Figure 16 FESTO Studio model, start segment....34
Figure 17 FESTO Studio model, middle segment....35
Figure 18 Cross conveyor....36
Figure 19 FESTO Studio model, end segment....36
Figure 20 Start segment of conveyor system....37
Figure 21 Middle segment of conveyor system....38
Figure 22 End segment of conveyor system....39
Figure 23 End lifter conveyor surface....43
Figure 24 Intermediate lifter....45
Figure 25 Cell m_ml_c1....46
Figure 26 Solution for cells with two inlets and two outlets....47
Figure 27 Simulation controllers setting....49
Figure 28 Manual Operation in FESTO CIROS Studio, I/O connection....50
Figure 29 SIMATIC manager....50
Figure 30 Interactions between segments....51
Figure 31 User interface for recipes....54

Figure 32 Cells numbering 55
Figure 33 Recipes loading 56
Figure 34 Pallet-position data block 57
Figure 35 Function logic for updating pallet position 58
Figure 36 Simulation screenshot 62
Figure 37 KPI framework [Rakar et al, 2004] 64

List of Tables

Table 1 Classifications of manufacturing simulation [Smith, 2003] 14
Table 2 Petri net properties and their meanings [Zhou and Venkatesh, 1999] 21
Table 3 Model hierarchy description of CIROS Studio 26
Table 4 Feature comparison of selected simulation software [Salminen, 2010] 30
Table 5 Weighted feature comparison of selected simulation software [Salminen, 2010] 31
Table 6 I/O descriptions of actuator mechanism 40
Table 7 I/O descriptions for sensor mechanism 41
Table 8 Decision points and corresponding inputs 53
Table 9 Recipe composition and execution time for single recipe 61
Table 10 Recipe relation 62
Table 11 Quantified simulation results of multiple recipes 63

1. INTRODUCTION

As one of the most tremendous infusive technology nowadays, simulation holds stupendous promise all over the manufacturing industry. From product development, prototype design to facility planning, mass production, each phase inside a manufacturing enterprise involves modelling and simulation. The more widespread simulation technology becomes, the more comprehensive operation simulation methodology should provide to users.

Meanwhile, integration between simulation models and real production lines tends to be crucial since it occupies relatively big proportion of monetary input. How to overcome the limitation of existing simulation methods falls into a significant research topic in manufacturing industry.

1.1. Scope

The simulation of production systems plays an important role in assessment of system performance. A visualization of simulation models simplifies the understanding of ongoing processes in the system. Possible integration of the simulation models with the production line could provide on-line monitoring.

In the current thesis work, different simulation strategies for the production lines are going to be evaluated. Based on FESTO CIROS Studio, a pallet-based assembly line is going to be modelled. According to the model, the real system performance should be assessed. Quantified measurements are tended to be collected from simulation model to determine related parameters like system throughput, how many pallets can be handled at the same time. Some key performance indicators (KPI) of production line should also be retrieved from the model.

1.2. Outlines

This thesis starts with a background research of modelling and simulation approaches that have been implemented in manufacturing industry. A survey into the domain of 3D simulation and modelling approaches will be discussed in chapter 2.

In chapter 3, 3D simulation model built in a FESTO CIROS Studio will be introduced based on the assembly system provided. How this model is integrated with Flexlink production line and based on what mechanism it was completed will also be covered. Simple user interface was established for determining recipes while the pallet loading for processing. For the purpose of handling multiple pallets at the same time, they were distinguished from each other in order to mark which path they should be routed.

Led by the creation of simulation model, a logic structure based on Siemens Simatic STEP 7 was implemented into the model and will be introduced in chapter 4. By inserting virtual programmable logic controller (PLC) into simulation, connections between production simulation and operation controller were fully established. I/O signals from production line (sensors and actuators embedded) were mapping to PLC control signals, and a program based on FBD and STL (Statement List) language was built to fulfil the pallet transportation routes.

By running the simulation model, production related assessment result will be measured and calculated quantified, and these parameters will be demonstrated in chapter 5. Also the quality of measurements and performance will also be evaluated in this section. System analyses will be introduced as well based by means of KPI.

Chapter 6 concludes this thesis. Evaluation of FESTO CIROS Studio will be analyzed based on its performance during current thesis topic. Possible future work directions and extended applications of the current simulation result in manufacturing industry will be discussed.

2. BACKGROUND RESEARCH OF SIMULATION

Jerry Banks defines simulation as the imitation of the operation of a real-world process or system over time [Banks, 1998]. In industrial field, manufacturing represents one of the most important applications of simulation.

2.1. Introduction to simulation

Simulation is one of the most powerful analysis tools available to those responsible for the design and operation of complex process or systems [Shannon, 1992].

It has been widely applied into various fields: computer systems, manufacturing industry, business analysis, military use, ecology, social studies, and biosciences and so on. The reason that simulation technology is so well adopted is the gap between objective reality and subjective perception. Figure 1 shows a simple example of simulation employment in automobile industry.

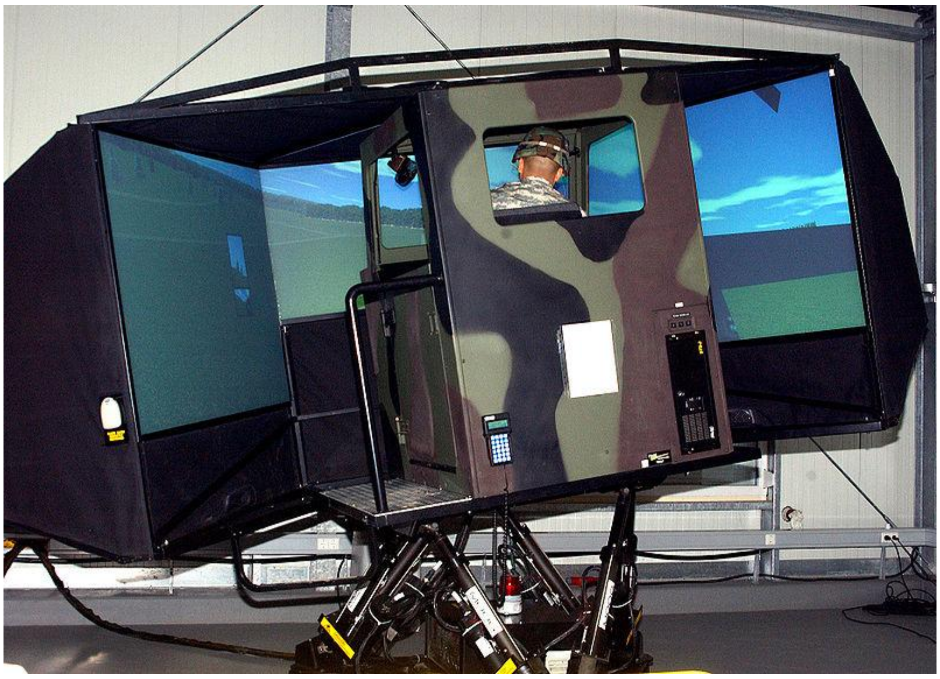


Figure 1 Vehicle simulator

When there is too much risk taking a new step, it is always a safe choice to experiment beforehand. This risk includes time issue, monetary issue, and safety issue and so on.

Thus, modelling and simulation technology holds tremendous promise of reducing cost, avoiding risk and increasing rate of success.

However, simulation can be utilized not only before producing real system but also at the same time of system carry-out. To simulate is essentially to duplicate the system or process for simplifying performance monitoring and analysis. From this perspective, simulation solves efficiently the problem that some huge-scaled systems with multiple types of input and output are hard to be evaluated by collecting real data resources. For example, it is not possible to calculate the exact rate of people who suffers diabetes over a country. But the result can be computed by taking a small sample of a group of people first, then approximate the ratio by mathematics method. This is one kind of simulation applying in public health.

2.1.1. Pros and Cons

To understand simulation, it is important to realize that it is not omnipotent for every case. Like all the other techniques, it still has two-side stories [Shannon, 1992].

For the bright side, firstly, simulation is an appropriate extending tool. It does not cause any interrupt to the existing system while it is on-going. Relatively less energy and other resources are needed for carrying out simulation process. And also it is a good way for exploring new policies and extending process procedures.

Secondly, simulation is a descent testing tool. It can be used for evaluating, such as lay-out design, hardware/software design, information and communication systems and so on, before being committed into the real system.

Thirdly, simulation is a diagnosing tool. By simulating real system performance, certain abnormality and errors can be found before actual implementation into real system. Meanwhile the cause of these abnormal phenomena can be diagnosed to decrease un-necessary capital expenses, time wasting and human resources, etc. For a positive result, the rate of feasibility is accordingly raised.

Fourthly, simulation is a good tool for controlling time during testing duration. In real case, it is not possible to observe long time system performance in short time, vice versa. While in simulation process, it is easy to control the speed of running model. Thus system performance trend can be estimated for long-time decision making and short time motion can be slowed down for detailed analysis.

Fifthly, simulation is a convenient analyzing tool. It helps to gain insight into the system and investigate the variables that matters to the real system without putting it to take risk. And it is also possible to analyze the interactions between different parameters and how they affect the performance of the entire system.

Sixthly, simulation is quite efficient for detecting bottleneck in material information and product flows. The situations such as equipment starving and blocking should be reported in simulation process. Simulation also identifies error information when other type of abnormality occurs.

Seventhly, simulation can be used as a verifying tool. Certain conceptions and suspecting cases, which gained from designing process, can be verified during simulation. The differences between how it is thought to be and how it really is can be revealed separately. Hypothesis can be tested to be applicable or not in advance.

Eighthly, after all, simulation is an experiment tool that answers to lots of “what if” questions. Before taking into real commitment, it is very common that designers hold limited knowledge about the actual system. By means of experiment allows us to testify all the suspects occurred during conceptual phase and recognize more factors about the simulated objects. More importantly, simulation provides a wide platform to try out different thoughts with no harm to the real, and in many cases, expensive system.

Even though simulation brought us these many conveniences, it still has some limitations as followings.

Model building is a subjective work that varies from individuals. The quality of analysis depends on the quality of the model and skill of the modeller [Shannon, 1992]. Generally, the more sophisticated and experienced modeller is the more comprehensive the model is to be.

Simulation results are sometimes hard to interpret. Since the simulation model is made from capturing randomness from production process, it is sometimes hard to identify if the simulation result is observed from a significant relationship from system or just a random occurrence built into the model [Shannon, 1992].

Simulation analysis can be a time consuming and expensive process. An adequate analysis may not be feasible within the time and/or resources available and a quick estimate using analytical methods may be preferable [Shannon, 1992].

Besides, the integration of simulation study and real production line may cost a fortune sometimes. Different application requires dedicated interface, which may cause complication of data transforming and/or protocol exchange, etc.

2.1.2. Simulation process

As demonstrated before, the purpose of applying simulation technology so tightly to other domains is that it is a logical system helping to solve technical problems. Therefore, to build a good simulation requires systematic procedures. There is one typical simulation process methodology concluded by Shannon (1992) is illustrated in Figure 2.

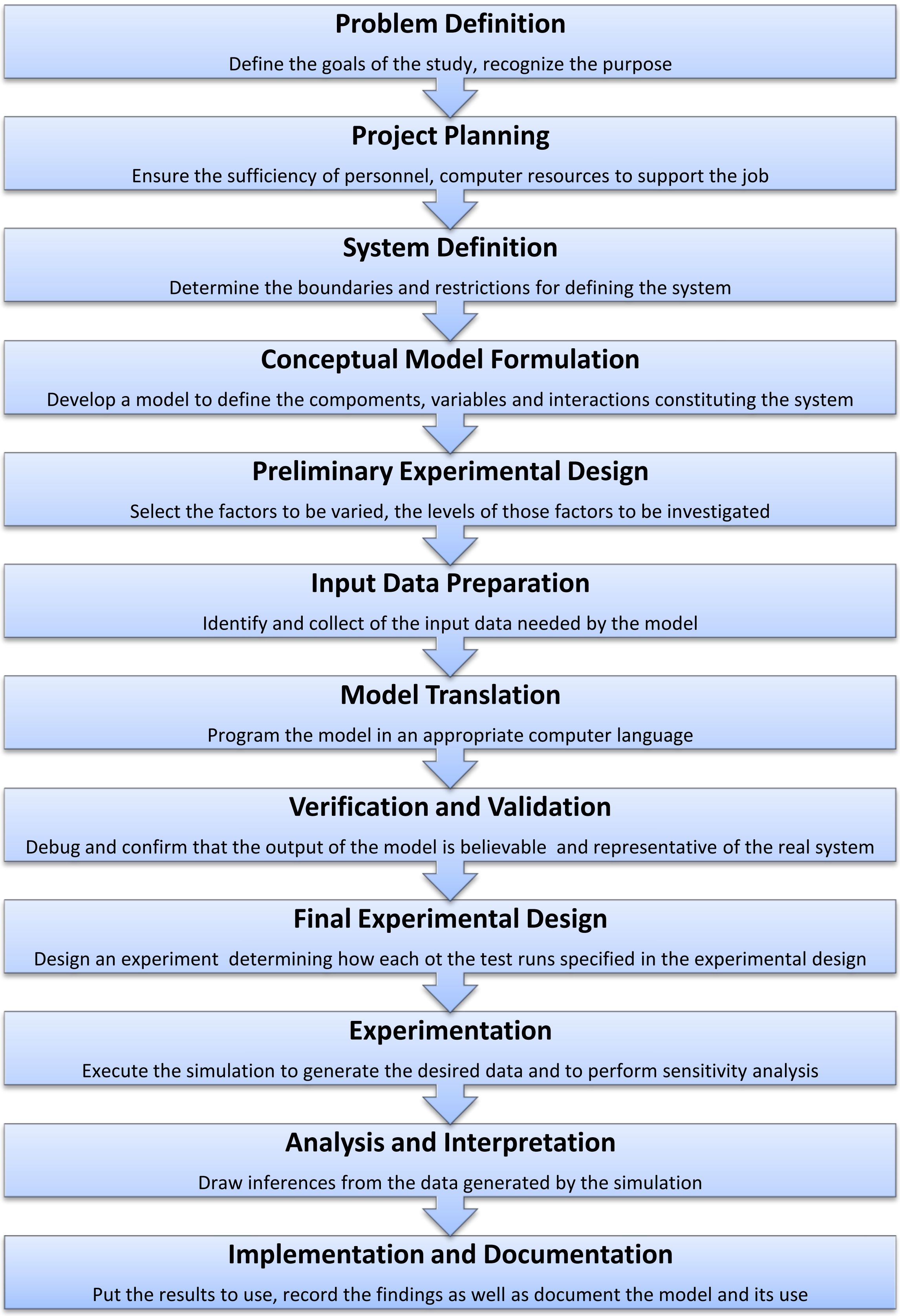

Problem Definition
Define the goals of the study, recognize the purpose
Project Planning
Ensure the sufficiency of personnel, computer resources to support the job
System Definition
Determine the boundaries and restrictions for defining the system
Conceptual Model Formulation
Develop a model to define the compoments, variables and interactions constituting the system
Preliminary Experimental Design
Select the factors to be varied, the levels of those factors to be investigated
Input Data Preparation
Identify and collect of the input data needed by the model
Model Translation
Program the model in an appropriate computer language
Verification and Validation
Debug and confirm that the output of the model is believable and representative of the real system
Final Experimental Design
Design an experiment determining how each ot the test runs specified in the experimental design
Experimentation
Execute the simulation to generate the desired data and to perform sensitivity analysis
Analysis and Interpretation
Draw inferences from the data generated by the simulation
Implementation and Documentation
Put the results to use, record the findings as well as document the model and its use

Figure 2 Process of simulation

2.2. Manufacturing simulation

The implementation of simulation in manufacturing system has been a hot topic for decades. As one of the most crucial techniques applied, simulation is a strong tool in general system analysis domain and performance evaluation in manufacturing system and operation design.

2.2.1. Benefits

Manufacturing simulation focuses on modelling the behaviour of manufacturing organizations, processes, and systems [McLean and Leong, 2001]. It can be used to:

- Model "as-is" and "to-be" manufacturing and support operations from the supply chain level down to the shop floor
- Evaluate the manufacturability of new product designs
- Support the development and validation of process data for new products
- Assist in the engineering of new production systems and processes
- Evaluate their impact on overall business performance
- Evaluate resource allocation and scheduling alternatives
- Analyze layouts and flow of materials within production areas, lines, and workstations
- Perform capacity planning analyses
- Determine production and material handling resource requirements
- Train production and support staff on systems and processes
- Develop metrics to allow the comparison of predicted performance against "best in class" benchmarks to support continuous improvement of manufacturing operations

Simulation models are built to support decisions regarding investment in new technology, expansion of production capabilities, modelling of supplier relationships, material management, human resources, and so forth [McLean and Leong, 2001].

2.2.2. Classification

Smith (2003) has divided the application of manufacturing simulation into three main classes, which are manufacturing system design, manufacturing system operation, and simulation language/package development for manufacturing system application [Smith, 2003]. Table 1 illustrates these three classifications and the major sub subjects in each division.

Table 1 Classifications of manufacturing simulation [Smith, 2003]

Class	**Sub subjects**

Manufacturing System Design	General system design and facility design/layout
	Material handling system design
	Cellular manufacturing system design
	Flexible manufacturing system design
Manufacturing System Operation	Operations planning and scheduling
	Real-time control
	Operating policies
	Performance analysis
Simulation Language/Package Development	

Manufacturing system design involves making long-term decisions [Smith, 2003], such as system layout, capacity or configuration design. In this category, systems are simulated macro. Once the simulation is finished, it may affect for long, time unit starting with weeks, months even for years.

On the other hand, manufacturing system operation involves on a much shorter time schedule, which means that the model is generally used (and reused) much more often and simulation run time is a more significant factor than the previous category [Smith, 2003]. Subjects like performance analysis and real-time control require frequent update as the collecting data fluctuates all the time. It may lose its power of reference when the information of the real system is obsolete.

2.3. Flexible Manufacturing System (FMS)

The essence of a Flexible Manufacturing System is a self-contained grouping of machinery that can perform all the operations, including transportation from one machine to another and/or performance under computer control with minimal human intervention, required in the manufacture of a number of parts with similar processing requirements [Young and Greene, 1986].

The concept of Flexible Manufacturing System is composed of the ideas of decision-making support system and adapting to changing environment. The system is designed to provide high productivity and flexible production capability.

The purpose of FMS is to realise flexibility in several areas inside manufacturing industry: machine flexibility, process flexibility, product flexibility, routing flexibility, volume flexibility, expansion flexibility, process sequence flexibility and production flexibility [Yilmaz and Davis, 1987].

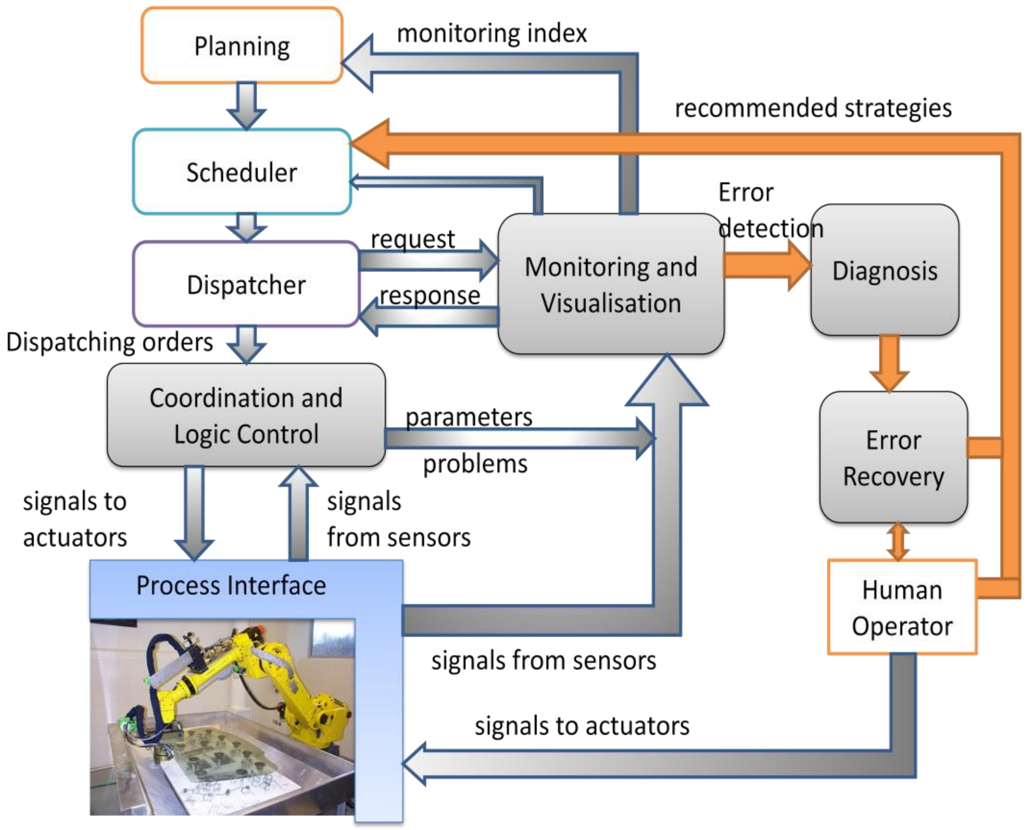


Figure 3 Major components of FMS and their relationships [Colombo, 2010]

Figure 3 indicates the overview of major functions of an FMS and the relationship between components can be summarized as following:

- The Decision Support System (DSS) consists of scheduler and dispatcher. It generates detailed scheduling tasks following the information from planning section. Decision Support System sends dispatching orders to downstream controllers. At the same time, DSS requests performance information from monitoring and visualisation system as the reference data in order to make self-developed decision for improving next decision.
- After receiving dispatching orders from DSS, coordination and logic control section translates these orders into detailed tasks to actuators and sends signals to each of them. Meanwhile it collects signals from sensors through process interface of the FMS. Then this section analyzes collected data, interprets into valuable production information, such as states of resources, error messages, problems to be solved, process parameters and so on, and deliver them all towards monitoring and visualisation section.
- Monitoring and visualisation sections plays as a bridge in the whole system, collecting data from all levels participated in the production activity and generating abnormality information to diagnosis centre. Simulation technology is one powerful tool in this section. It listens to not only the controller but also feedback signals from hardware components (sensors). This is necessary for building the database

for modelling and simulation. As the output, this section responses to DSS real-time information and also provides monitoring index to planning centre, offering objective reference and helping to establish next plan.

- Diagnosis section receives error detection from monitoring and visualisation, works out recovery solutions, and then information flows to human operator to give instruction of repairing advices. Hardware components (actuators) are fixed to overcome errors and limitations. By diagnosing procedure, experience of dealing some problem situation is gained, and recommended strategies is offered as an output into DSS, helping to perfect next decision.

2.4. Multi-Agent Simulation (MAS)

A multi-agent system can be defined as a set of agents represent the objects of a system, capable of interacting, in order to achieve their individual goals, when they have not enough knowledge and/or skills to achieve individually their objectives [Leitao, 2009].

A suitable definition, originated from the definition of multi-agent system, for agent is: “An autonomous component that represents physical or logical objects in the system, capable to act in order to achieve its goals, and being able to interact with other agents, when it does not possess knowledge and skills to reach alone its objectives” [Leitao, 2009]. For example, people, organizations, social insects, robots can all be considered as agents with their own goals and behaviours.

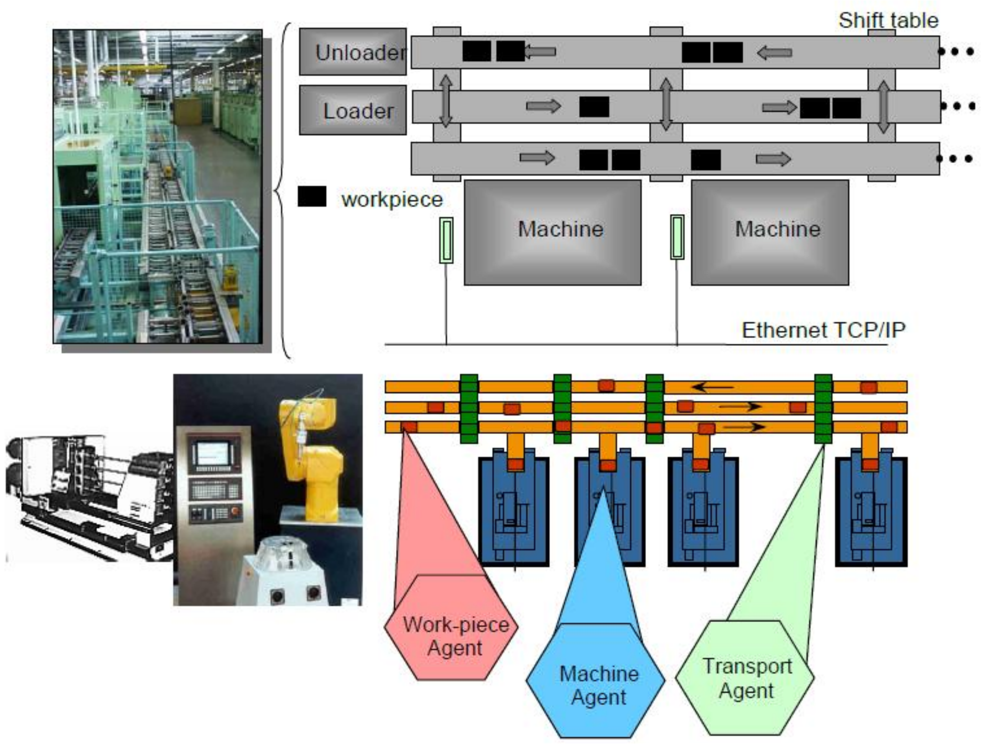


Figure 4 Agent-based Information Technology Fusion in Mechatronics [Colombo, 2010]

In a manufacturing system, three typical agents are implemented, as indicated in Figure 4, work-piece (pallet) agent, machine agent and transport agent. An example of these three agents' interaction is given in Figure 5.

Three main scenarios can be summarized from Figure 5.

- Work-piece agent sends out requests to ask for a machine to be operated;
- All three agents that represents machines reply to the work-piece agent and report their status, and only machine agent #3 responses positively;
- After identifying departure and destination location information, work-piece agent starts to negotiate with transport agent, transport agent plans for the route and transportation system is in charge of implementing orders of leading work-piece to the position of machine#3.

It is important to recognize that the control system is independent of the number of machines in the system and it does not notice the introduction of new machines or existing machines removing; also, agents that represent several machines are upgraded using same customized development software, according to their type, skills and behaviour [Leitao, 2009].

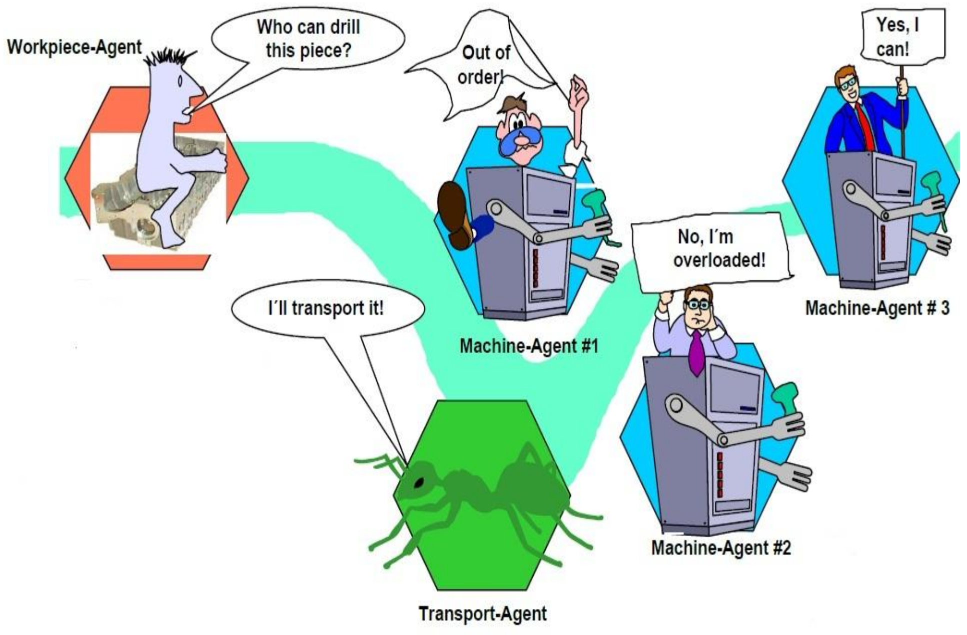


Figure 5 Example of agent interactions in manufacturing control [Colombo, 2010]

The agent-based control system should be integrated to the commensurate industrial automation system as to emulate real-time operation. To realize machine autonomy, Computer Numerical Control (CNC) machines are implemented as the machine agent; for the same purpose, Programmable Logic Controller (PLC) for transportation system and Industrial Personal Computer (IPC) for work-piece agent.

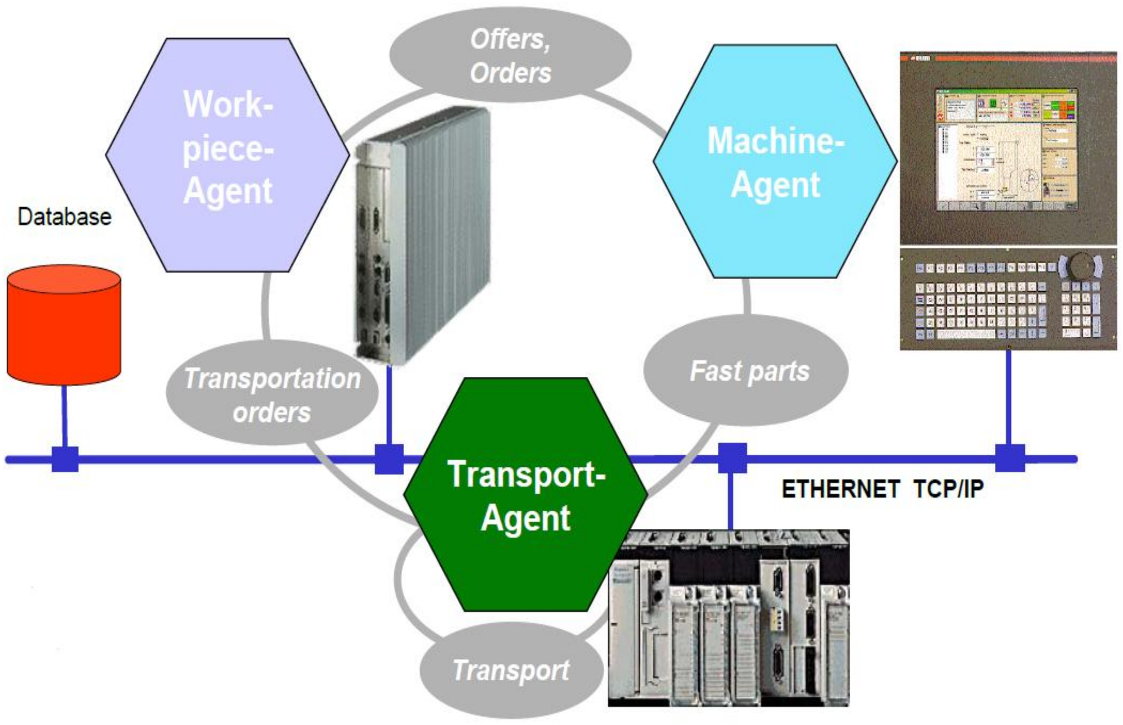


Figure 6 Collaborative production automation architecture [Colombo, 2010]

To enable the communication and information synchronization between all kinds of agents, a universal database is needed as to data storage for supporting production related decisions. Physical connections can be built by means of standardized physical link layer methodology, such as Ethernet, Modbus and so on. An agent-based manufacturing architecture based on collaboration is illustrated in Figure 6.

2.5. Petri Net

Petri Net, as simply defined, is a graphical and mathematical modelling tool. It is a promising methodology for describing and studying information process systems.

Due to the generality and permissiveness inherent of Petri Net, it can be applied in many areas and systems. Two successful application fields are communication protocols and performance evaluation, and other promising areas of applications include modelling and analysis of distributed-software systems, distributed-database system, concurrent and parallel programs, flexible manufacturing/industrial control systems, discrete-event systems, dataflow computing systems, fault-tolerant systems and so on [Murata, 1989].

2.5.1. Formal definition of Petri Net

A Petri Net is a 5-tuple, $PN = (P, T, F, W, M_0)$ where:

$P = \{p_1, p_2, \dots, p_m\}$ is a finite set of places,

$T = \{t_1, t_2, \dots, t_n\}$ is a finite set of transitions,

$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation),

$W: F \rightarrow \{1,2,3, \ldots\}$ is a weight function,

$M_0: P \rightarrow \{0,1,2,3, \ldots\}$ is the initial marking.

$P \cap T = \emptyset$ and$P \cup T \neq \emptyset$.

A Petri net structure $N = (P, T, F, W)$ without any specific initial marking is denoted by$N$. And a Petri Net with the given initial marking is denoted by(N, M_0).

Places, transitions and the interconnections between places and transitions can be symbolized as Figure 7. Introduction tokens into places and observation of their flow path help one to understand discrete-event behaviour of PN as well as the modelled system.

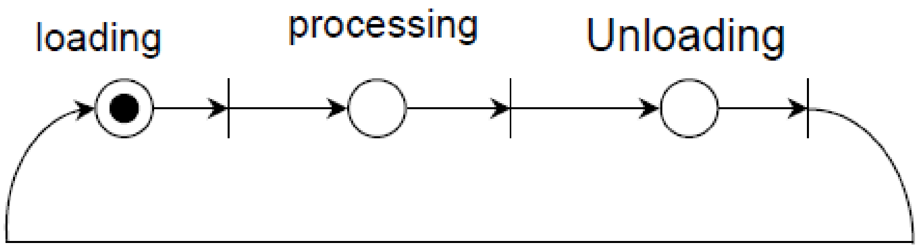


Figure 7 PN graphic example

2.5.2. Properties of Petri Net

Like every other mathematical methodology, Petri net owns several properties which enable users to identify the presence or absence of the functional properties of the system. Two types of properties can be distinguished, behavioural and structural ones [Zhou and Venkatesh, 1999]. The behavioural properties are those which depend on the initial state or marking of a PN. On the other hand, structural properties do not depend on the initial status of a PN but PN topology or structure only.

Murata (1989) classified behavioural properties into eight sorts: reachability, boundedness, liveness, reversibility and home state, coverability, persistence, synchronic distance, and fairness.

- **Reachability**

A marking M_n is said to be *reachable* from a marking M_0 if there exists a sequence of firings that transforms M_0 toM_n. A firing or occurrence sequence is denoted by $\sigma = M_0\ t_1\ M_1\ t_2\ M_2\ \cdots\ t_n\ M_n$ or simply$\sigma = t_1\ t_2\ \cdots\ t_n$.

- **Boundedness and Safeness**

A Petri net (N, M_0) is said to be *k-bounded* or simply *bounded* if the number of tokens in each place does not exceed a finite number k for any marking reachable fromM_0.

A Petri net (N, M_0) is said to be *safe* if it is 1-bounded.

- **Liveness**

A Petri net (N, M_0) is said to be *live* (or equivalently M_0 is said to be a *live marking* for N) if, no matter what marking has been reached from M_0, it is possible to ultimately fire *any* transition of the net by progressing through some further firing sequence.

The implications of these properties in manufacturing industries are summarized in the following table.

Table 2 Petri net properties and their meanings [Zhou and Venkatesh, 1999]

PN Properties	Meanings in the Modelled Manufacturing System
Reachability	A certain state can be reached from the initial conditions
Safeness	Availability of a single resource; or no request to start an on-going process
Boundedness	No capacity (of, e.g., buffer, storage area, and workstation) overflow
Liveness	Freedom from deadlock and guarantee the possibility of a modelled event, operation, process or activity to be on-going

2.5.3. PN Application

It is well known that Petri net technology has been widely adopted in various industrial fields including manufacturing fashion. There are three main topics that have been applied with PN in factory automation, and they are summarized as followings.

2.5.3.1 Manufacturing, production and scheduling systems

Petri net technology can be used to model production events. Regarding the manufacture resources as fixed entity and production task as mobile entity, a sample PN model can be established as shown in Figure 8.

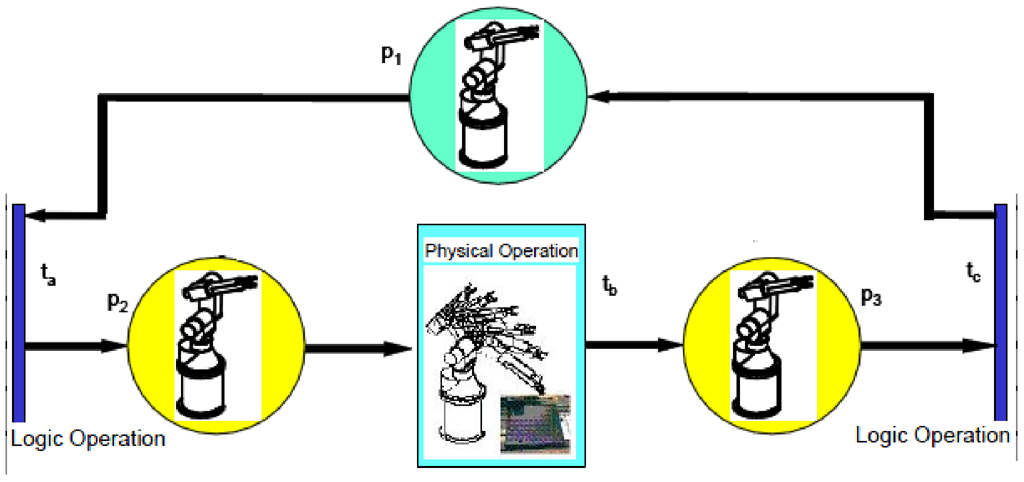


Figure 8 Sample PN model for production events [Colombo, 2010]

In the model, t_b represents the robot processing operation, P_1 represents the state that resource is free, P_2 represents the state that resource is busy before operation, P_3 represents the state that resource is busy after operation, and t_a and t_c are the transitions between idle and busy state.

For the purpose of scheduling system in manufacturing process, Lloyd et al. (1995) introduced a modified branch & bound methodology for scheduling algorithm. Integrated with Petri net modelling and reachability generating, the proposed approach was proved to show some improvements than previous work.

2.5.3.2 Sequence controller (Programmable Logic Controller, PLC)

A PLC is a digital computer used for control automation operation. The first development of PLC was to replace hard-wired control equipment. Nowadays, PLCs have been widely employed in automation areas from discrete manufacturing plants to continuous processes. Usually, PLC can be programmed using five standard programming languages: function block diagram (FBD), structured test (ST), ladder diagram (LD), instruction list (IL) and sequential function chart (SFC).

Minas and Frey (2002) proposed a special type of Petri net, the Signal Interpreted Petri Net (SIPN) in their study. Comparing to conventional Petri net modelling, signals are introduced as the symbolism for influence caused by environment changing, which, in PN word, are the conditions for firing transitions. In this way, several transitions can be fired simultaneously due to signals changing. SIPN allows unstable states to exist cue to its dynamics property, and certain transitions can be fired at the same time until a stable stated is reached. This new language was proved, in a university course experiment, to be applied easier than standard PLC languages. During the formal correctness and transparency analyses, SIPN also showed improvements to the design process.

An extended example of Petri net application in control principle was placed for Discrete-Event Control (DEC), and this methodology is called PNDEC (Petri Net based Discrete-Event Controller) [Korotkin et al, 2010]. The main idea for PNDEC is to assign the readings of sensors and actuators from the Discrete Event Systems (DES) as input signals of the controller and output of controllers as control actions back to DES. Using PN to describe control logic, a set of input reading combinations are applied as firing conditions for PN model. A sample PNDEC integrated with FBD is shown in Figure 9.

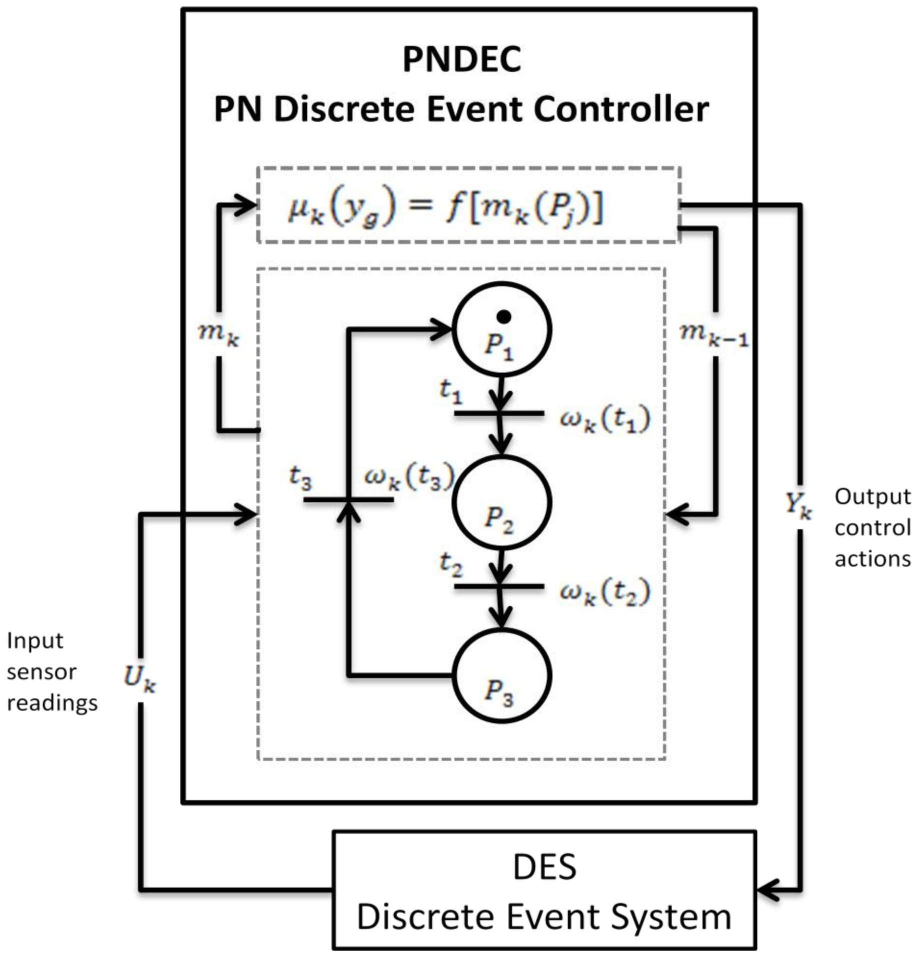


Figure 9 Block diagram of PNDEC [Korotkin et al, 2010]

2.5.3.3 Communication protocols and networks

A generalized timed Petri net representation was defined by Zhu and Denton (1988) to model entity behaviours in communication networking. Timed Petri nets are distinguished from conventional PN by introducing time variables. The reason of choosing timed Petri net for modelling communication protocols is that each level of protocols is built based on real-time property.

In their study [Zhu and Denton, 1988], three basic phenomena example in communication technology were given, dealing with transmission error, timer and communication protocol (by specifying sender and receiver behaviours).

2.5.4. Advantages of PN in production simulation

As a graphical modelling tool, Petri net provides users a unified design approach for discrete event system. Other than this, there are still many advantages that make Petri net a promising tool in production automation field.

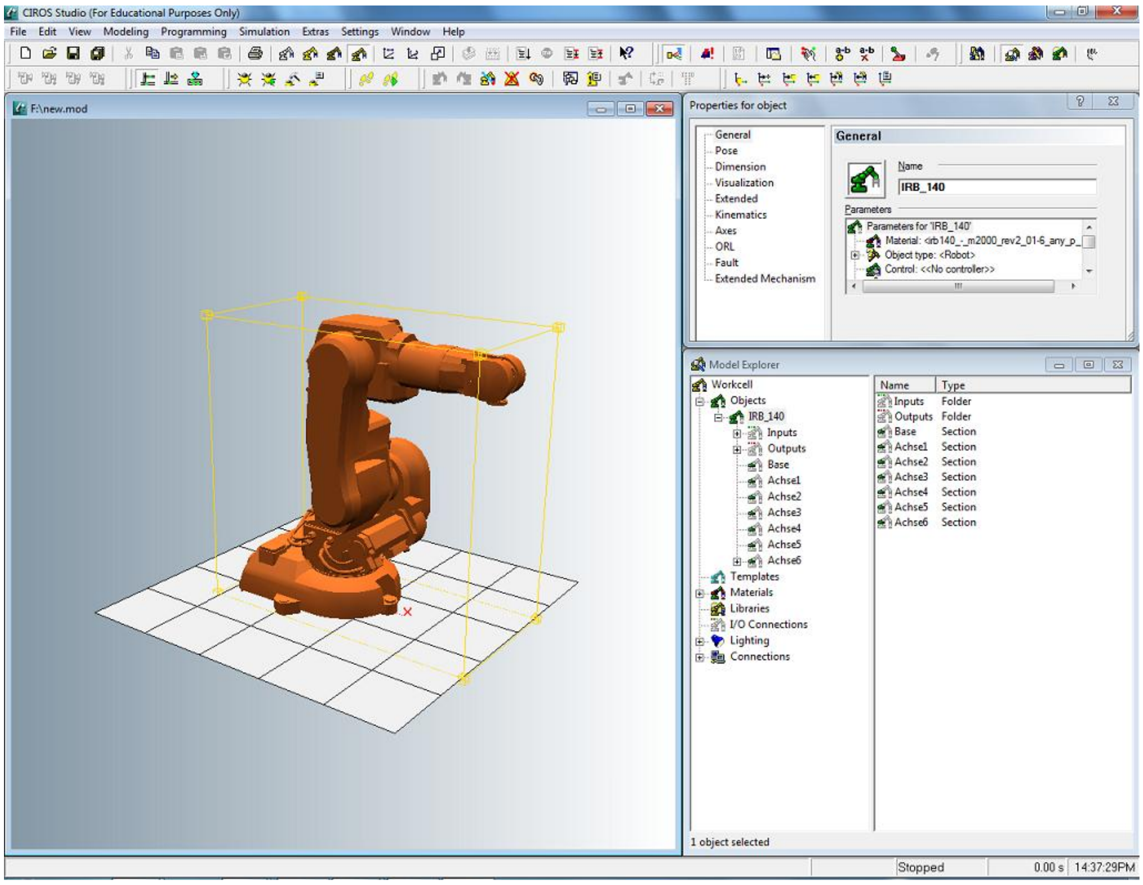
1. Ease of modelling characteristics of a complex industrial system: concurrency, asynchronous and synchronous features, conflicts, mutual exclusion, precedence relations, non-determinism, and system deadlocks [Zhou and Venkatesh, 1999]. Petri nets models offer excellent visualization of system dependencies. They focus on local information rather global one. Top-down (stepwise refinement) design, bottom-up (modular composition) design, and hybrid methods can be applied to design and construction of Petri nets models.
2. Ability to generate supervisory control code directly from the graphical Petri net representation [Zhou and Venkatesh, 1999]. A Petri net execution algorithm can also be constructed for real-time implementation using either Programmable Logic Controllers (PLC) or computers.
3. Ability to check the system for undesirable properties such as deadlock and capacity overflow and to validate code by mathematically based computer analysis - no time-consuming simulations are required for many cases [Zhou and Venkatesh, 1999].
4. Performance analysis without simulation for many systems. Production rates, cycle time, resource utilization, reliability, and performability can be evaluated [Zhou and Venkatesh, 1999]. Bottleneck machines can be identified.
5. Discrete event simulation that can be driven from the model [Zhou and Venkatesh, 1999].
6. Status information that allows for real-time control, monitoring and error recovery of FMS [Zhou and Venkatesh, 1999].
7. Usefulness for scheduling because the Petri net model contains the system precedence relations among events, concurrent operations, appropriate synchronization, repetitive activities, and mutual exclusion of shared resources, as well as other constraints on system performance [Zhou and Venkatesh, 1999].

2.6. CAD/CAM tools

Integration of Computer-Aided Design and Computer Aided Manufacturing is a significant topic in industrial automation. It enables engineers to gain an insight preview of systems, helps to improve quality of products and optimize production time. Several 3D simulation tools are widely applied in the field, and some of them are introduced and compared in this section.

2.6.1. FESTO CIROS Studio

CIROS Studio is the universal 3D simulation system developed by FESTO Didactic, belonging to the CIROS Automation Suite. In Figure 10, there is the screenshot of CIROS Studio interface illustrated. And this software is chosen to create the 3D model for the given production line in current thesis.



General

Name: IRB_140

Parameters
Parameters for 'IRB_140'
Material: <irb140_-_m2000_rev2_01-6_any_p_
Object type: <Robot>
Control: <<No controller>>

Name	Type
Inputs	Folder
Outputs	Folder
Base	Section
Achse1	Section
Achse2	Section
Achse3	Section
Achse4	Section
Achse5	Section
Achse6	Section

1 object selected

Stopped 0.00 s 14:37:29PM

Figure 10 FESTO CIROS Studio screenshot

CIROS Studio, in a nutshell, enables users to create a detailed planned workcell or an entire production line, to simulate robots operations associated with controller behaviours (external or internal), to test the reachability of critical positions, and to observe production processes.

2.6.1.1 Modelling

In CIROS Studio, plenty of existing model libraries are provided for efficient modelling, materials, machineries, robots, controllers, and well-made mechanisms like sensors and conveyor belts. After choosing from model libraries, relevant properties and components of the object can be viewed in detail as well. Commensurate I/O configuration can also be found in well-made mechanisms, which is able to be controlled manually through manual operation tab. Signal changing is easy to observe, and connections between inputs and outputs can be established clearly in operation window.

It is also possible for the users to create 3D objects of their own. This enables the users to create their own libraries for needed job. Simple geometric shaping objects can be built by defining size related parameters. In current thesis, a 3D model of given production line is supposed to be created from scratch. During modelling, it is also very important to realise the model hierarchy, which contains the following element types in Table 3.

Table 3 Model hierarchy description of CIROS Studio

Icon	Element name	Hierarchy description	Example
	Objects	The highest unit in the element structure are the objects.	A robot is an object.
	Sections	Sections are assigned to objects. One degree-of-freedom can be associated to each section that is moveable relatively to the previous section.	Each joint of a robot is a section.
	Hulls	Hulls are assigned to sections and are responsible for the graphical representation.	A face, a box or a polyhedron are hulls.
	Gripper Points	An object needs a gripper point to grasp other objects. Gripper points are assigned to sections	At the flange of a robot, a gripper point is modelled.
	Grip Points	To be grasped by another object, an object needs a grip point. Grip points are assigned to sections.	A grip point is associated to a work piece that has to be grasped.

2.6.1.2 Programming

Workcell programming is based on the creation of position list in advance. A position list contains all the must-go points from the robot processing route. Each position point can be edited in properties menu by defining x, y, and z parameters.

After accepting a position list for robot, two programming languages can be applied to model robots behaviours, which are IRL (Industrial Robot Language) and Melfa Basic IV.

2.6.2. QUEST

QUEST (QUeueing Event Simulation Tool) is a well-known object-based, discrete event simulation tool. It belongs to a Delmia product family, Dassault Systems, which is aiming for digital manufacturing and production virtual design. Mastering QUEST allows manufacturers in any industry to define, plan, create, monitor and control all production processes virtually.

Modelling in QUEST is applied by means of creating elements, positioning them on the layout and defining relative parameters. Delmia QUEST provides a bunch of element classes and each element class possess an individual group of parameters which determine the outlook and behaviours of the model. Digital inputs and outputs have to be created and connected. Complicated functional logics and production processes need to be programmed in SCL or BCL language.

From users' perspective, Delmia QUEST provides a collaborative platform for industrial engineers, manufacturing engineers and management to develop and prove out best manufacturing flow practise. It allows users to build a simulation model from conception phase to implementation phase, adding design details as needed through the whole development process. The advantages of QUEST can be summarized as followings:

- Observe, interact and analysis of "what if" scenarios
- Import CAD and other data such as scheduling and routing
- Complete integration with other Delmia process planning and simulation solutions
- Identify bottlenecks
- Optimize labour and production schedules

2.6.3. Taylor Enterprise Dynamics

Taylor Enterprise Dynamics (Taylor ED) is object-oriented software system used to model, simulate, visualize, and monitor dynamic-flow process activities and systems [Nordgren, 2001]. It was developed by Incontrol Simulation Solutions, belongs to a product serial which is also aiming for solutions in other fields, Logistics, Airport, Transport, Warehouse, Plato, Education and so on.

The foundation of Taylor ED modelling is called "atom". An atom is an object with four dimensions (x, y, z, and time), and each atom can have a location, speed, and rotation (in x, y, and z) and dynamic behaviour over time [Nordgren, 2001]. The control logic of each atom is defined with a script language called 4Dscript which is similar to Basic.

To build a model, two general steps are determined. Starting model building, the atoms can be easily dragged out of the library into operation window. By right clicking on the atom, an input window containing general properties of the atom appears and users can edit, for example, the inter-arrival time field to customize each atom according to different requirements. Once the model is created, channels connecting atoms should be established and enabled. Each atom may contain multiple input and output channels, and the connections is successfully built when both input and output channels are open.

The model of Taylor ED can be viewed both in 2D and 3D version, which enable the users to view logical insights among atoms in 2D and visually simulate in 3D. An example of presentation of Taylor ED is shown in the figures below.

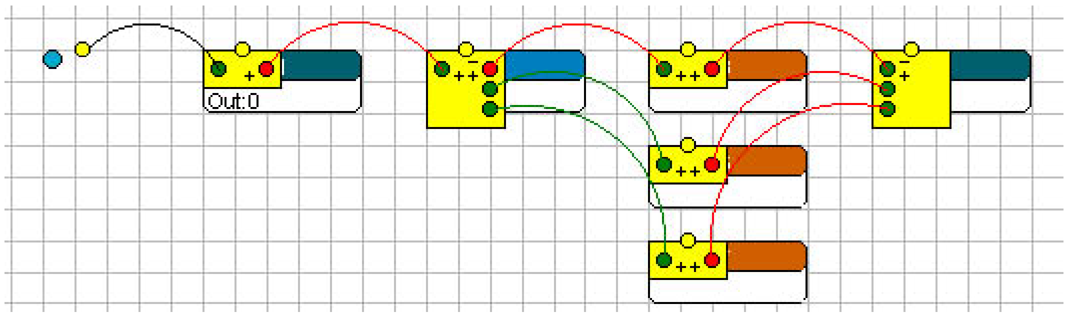


Figure 11 Taylor ED, 2D model with connected channels [Incontrol, 2011]

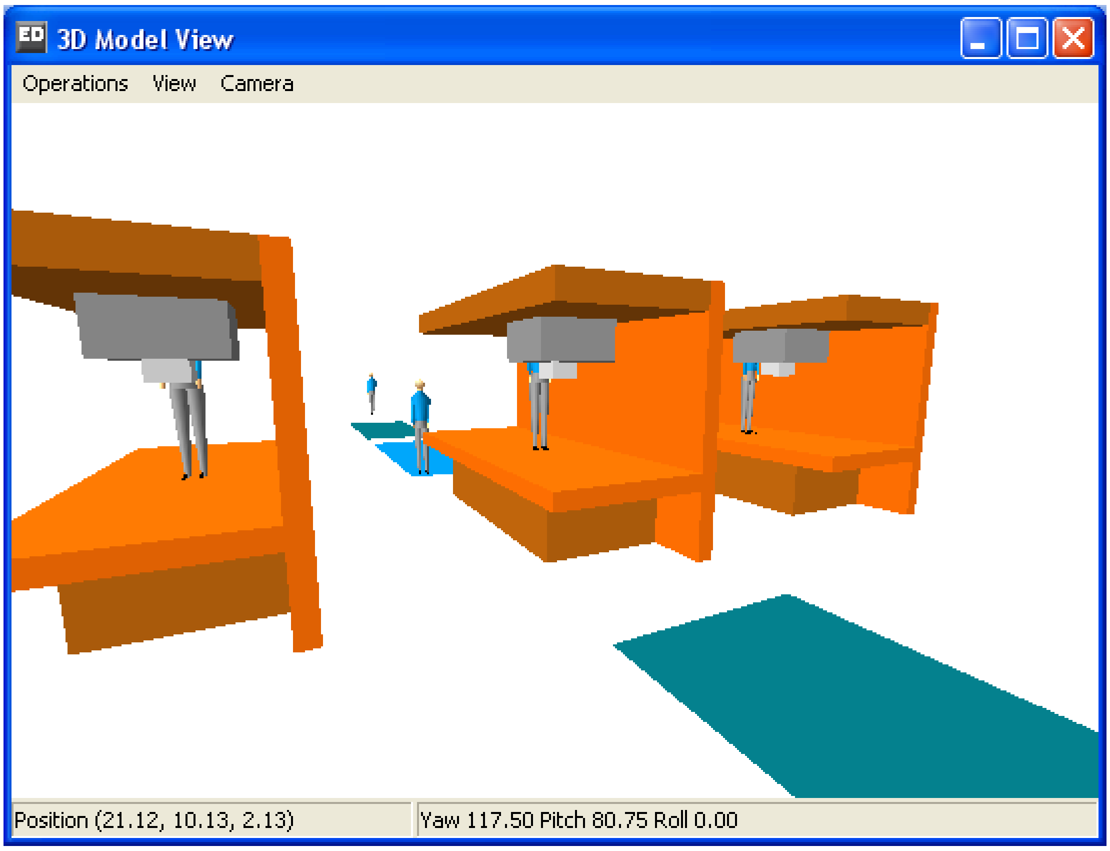

3D Model View
Operations
View
Camera
Position (21.12, 10.13, 2.13)
Yaw 117.50 Pitch 80.75 Roll 0.00

Figure 12 Taylor ED, 3D model of three counters [Incontrol, 2011]

2.6.4. Visual Components product family

3DRealize is component-based 3D software for production line simulation which was developed by Visual Components Oy. Visual Components was founded in 1999 as a joint venture with JOT Automation Oy, and later in 2001 became independent. It offers a suite of 3D software solution package including 3DCreate, 3DSimulate, 3DRealize R, and 3DRealize. These software can be viewed free of charge from Visual Components official website. User interface of 3DRealize user interface is illustrated in Figure 13.

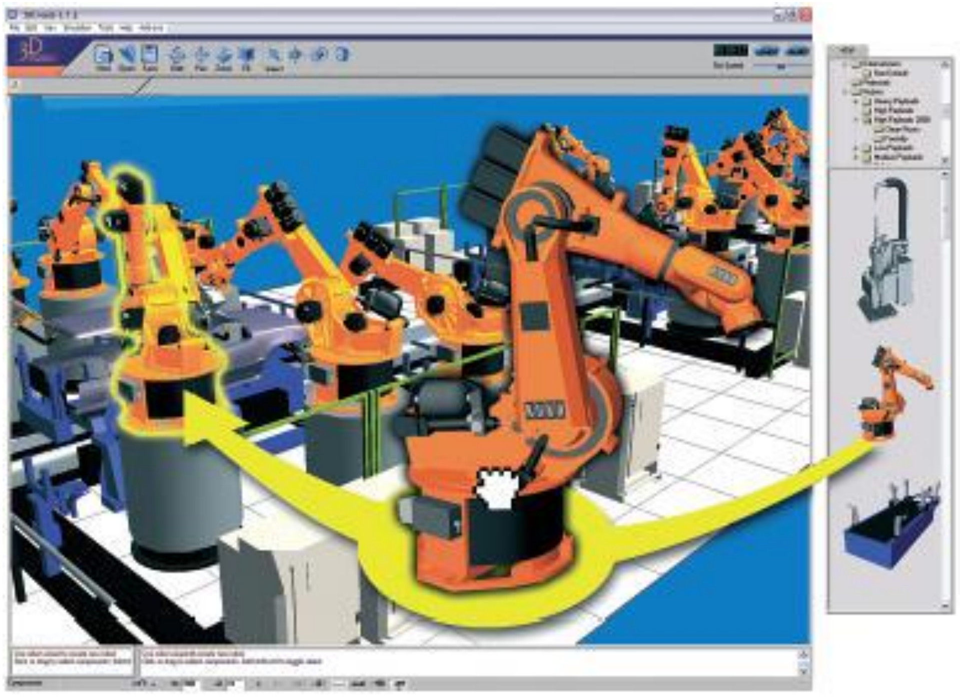


Figure 13 3DRealize interface [Visual Components]

3DRealize is a powerful tool to generate 3D production line models that actually run. With 3DRealize, one can easily import or modify any existing models and it can automatically recognize compatible equipments, which makes production line layout creation a piece of cake. Equipments can be dragged and dropped from side bars simply. After completing layout, some production indicators, such as energy consumption, energy efficiency, and throughput, can be evaluated, which benefit factory engineers with multiple alternatives of layout design, low risk of wrong investments, analysis of production plans and system performance in advance, and reducing unnecessary costs eventually. Simple operation benefits not only manufacturing engineers but also sales staffs. Visualization and presentation can be more vivid and convincing for the customers since design concept can be directly perceived through then sense.

Another major advantage of 3DRealize is that model files are relatively smaller than other 3D simulation software, usually less than 100kb. This factor enables engineers to send models via email, among layout designers, manufacturers and plant managers. Furthermore, engineers can share resources at within short time and participate in real-time discussions and communications. It also solves the time consuming issues caused by distant physical location of different staffs.

Meanwhile, Visual Components software suite also provides solutions for machine building, system integration, robot simulation, material handling and PLC add-ons.

2.6.5. Comparison

The finalists simulation tools compared in this section were proposed by Salminen (2010), which were discussed with key people at Flexlink Automation. These softwares somehow satisfy the demands of Flexlink modelling, which can be summarized as follows [Salminen, 2010].

- Capability to handle real production variables: physical lengths, speeds, accelerations and decelerations; utilization of a realistic plant layout for analyzing the effects on material handling equipment and labour
- Capability to allocate resources required for certain processes
- Possibility to use realistic movement paths
- Possibility to use automated storage and retrieval systems
- Allowance of utilization of Flexlink Automation's existing CAD and visualization tools, especially the already existing models and geometries
- Interactive 3D environment provided, where different line solutions can be visualized and studies
- Flexible, easy-to-use material handling modules provided
- Features or future possibilities for reporting, exporting, importing and database connectivity
- Compatible programming languages with the ones that has been already used by Flexlink Automation

For comparison purpose, impact factor was determined also by group discussion based on effect importance and user experience. The evaluation results are demonstrated in following tables.

Table 4 Feature comparison of selected simulation software [Salminen, 2010]

	3DRealize	TRAM PLB	Taylor ED	QUEST	Average	IMPACT FACTOR
Learning curve	5	4	3	2	3.5	4
Ease of Use	4	3	3	3	3.25	5
GUI	4	3	2	2	2.75	3
Graphics	5	5	2	3	3.75	5
Speed	2	4	4	5	3.75	3
Modularity	5	4	3	2	3.5	4
Plug and Play	5	4	3	1	3.25	4
AutoCAD connection	4	3	4	4	3.75	3

Statistics	1	1	3	5	2.5	2

Table 5 Weighted feature comparison of selected simulation software [Salminen, 2010]

	3DRealize	**TRAM PLB**	**Taylor ED**	**QUEST**	**Average**
Learning curve	20	16	12	8	14
Ease of Use	20	15	15	15	16.25
GUI	12	9	6	6	8.25
Graphics	25	25	40	15	18.75
Speed	6	12	12	15	11.25
Modularity	20	16	12	8	14
Plug and Play	20	16	12	4	13
AutoCAD connection	12	9	12	12	11.52
Statistics	2	2	6	10	5
TOTAL	137	120	97	93	111.75
POSITION	1	2	3	4	

Combined with the weighted parameters, these softwares can be compared to each other. The result shows that 3DRealize is more suitable for Flexlink Automation that others. TRAM PLB was tested faster and more tailor-made, on the other hand, 3DRealize benefits with easier use and seems to be more open for future development [Salminen, 2010].

3. CASE STUDY PRESENTATION

In order to generate performance analysis result of simulation model, an assembly line customized by Flexlink is introduced as the object to be investigated. The outlook of the real line is shown in Figure 14 and the I/O configuration can be found in Appendix 1. In this chapter, a 3D simulation is created based on FESTO CIROS Studio.

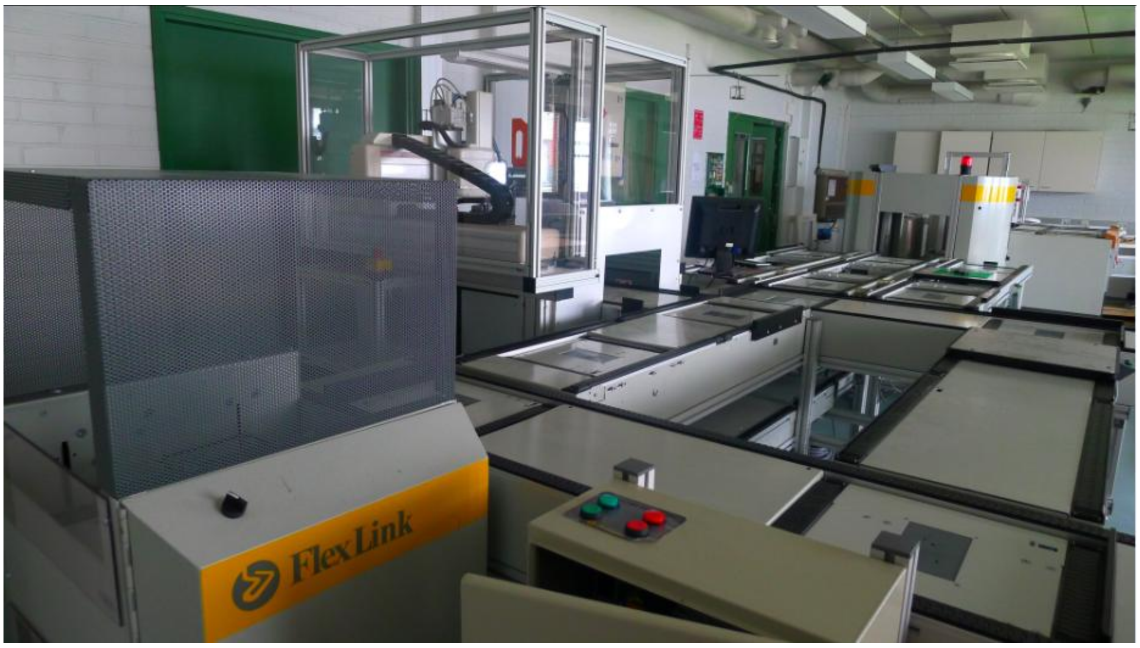


Figure 14 Flexlink production line

3.1. Layout of assembly line

A FESTO CIROS Studio model was built according to the actual measurements taken from assembly line. The top view of the whole line is illustrated in Figure 15. In this thesis, only the pallet transportation system is taken into simulation consideration, which means that the robot execution and manual operations are not included.

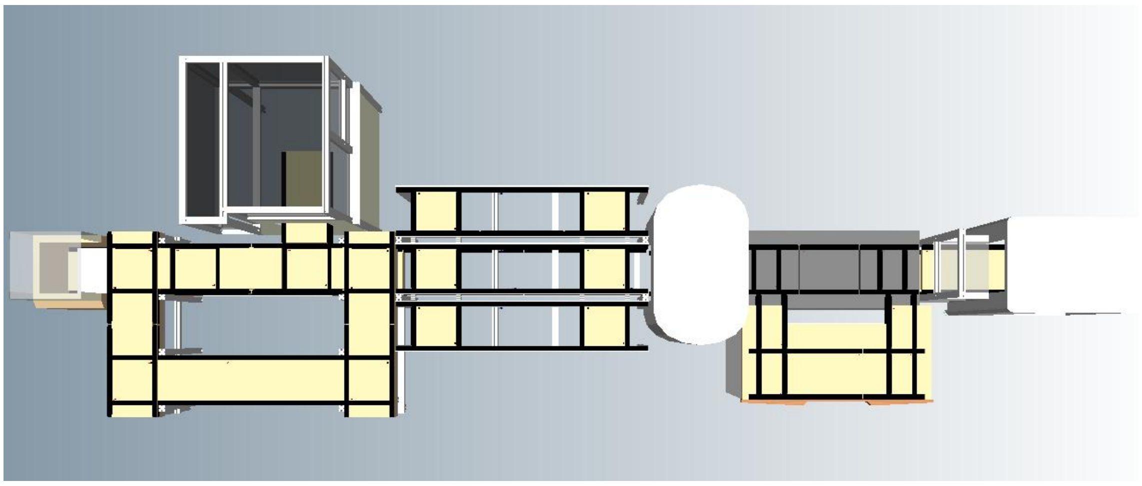


Figure 15 Top view of FESTO Studio model

The assembly line is composed of three sections: start segment, middle segment and end segment. There are totally two layers of conveyors: upper layer and lower layer. The pallet is a 400×400×50mm metal plane tray which supports the parts to be processed among different workcells. The pallet flows firstly on upper conveyor layer, starting from the start segment to middle segment and end segment, then translates to lower conveyor layer and returns back.

3.1.1. Start segment

Start segment (DAS Lite) is composed of one start lifter (5099EN-1HC), one manual workstation (5098EN-1HC), one customized robot cell and a portion of the mainline.

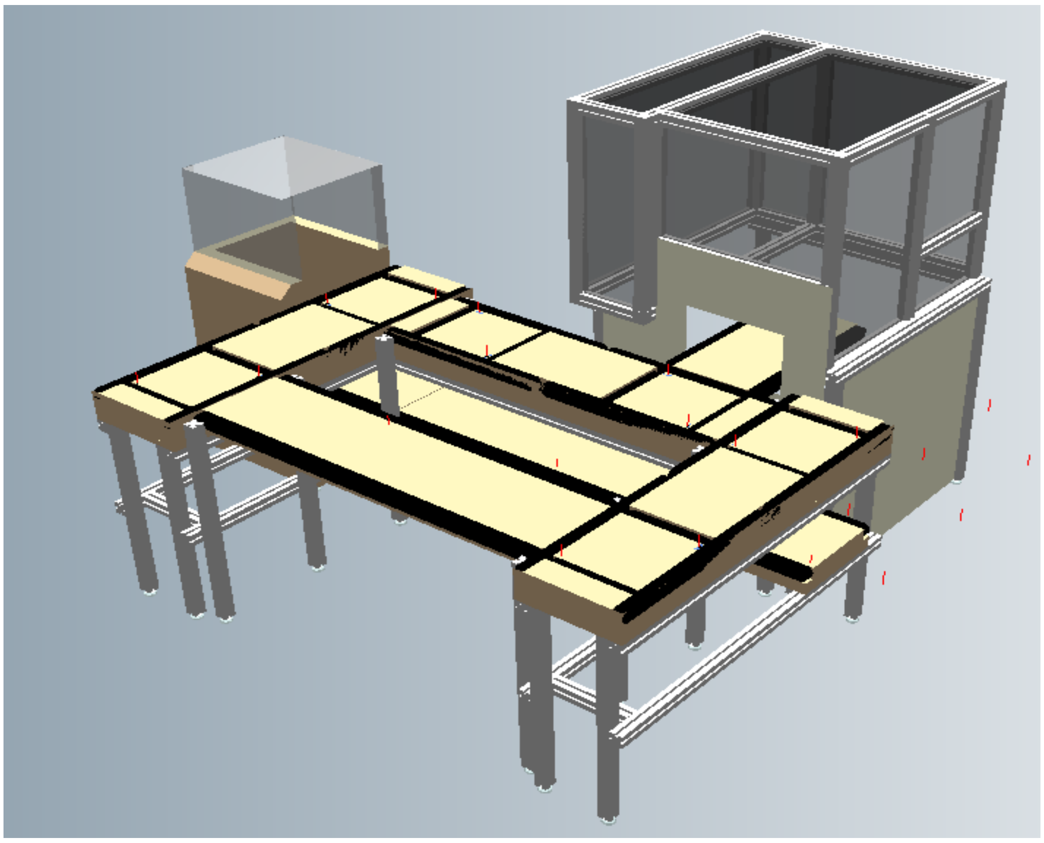


Figure 16 FESTO Studio model, start segment

Paired with the layout model, a S7 simulation controller model was inserted virtually as the internal PLC which was dedicated to the start segment. All compulsory connections between mechanism model and controllers can be established in "Manual Operation" window which can be found in "Modelling" menu. Therefore, two objects complete the layout model of start segment:

- StartSegment.mod
    a) StartLifter
    b) S_MainLine1 (as a portion of mainline, with two level of conveyors)
    c) S_MainLine2 (vertical line near start lifter)
    d) S_MainLine3 (vertical line near middle segment)
    e) S_RobotCell
    f) S_WorkStation
- S_SimulationController

3.1.2. Middle segment

Middle segment (DAS 30) includes two manual workstations, one intermediate lifter and a portion of the mainline.

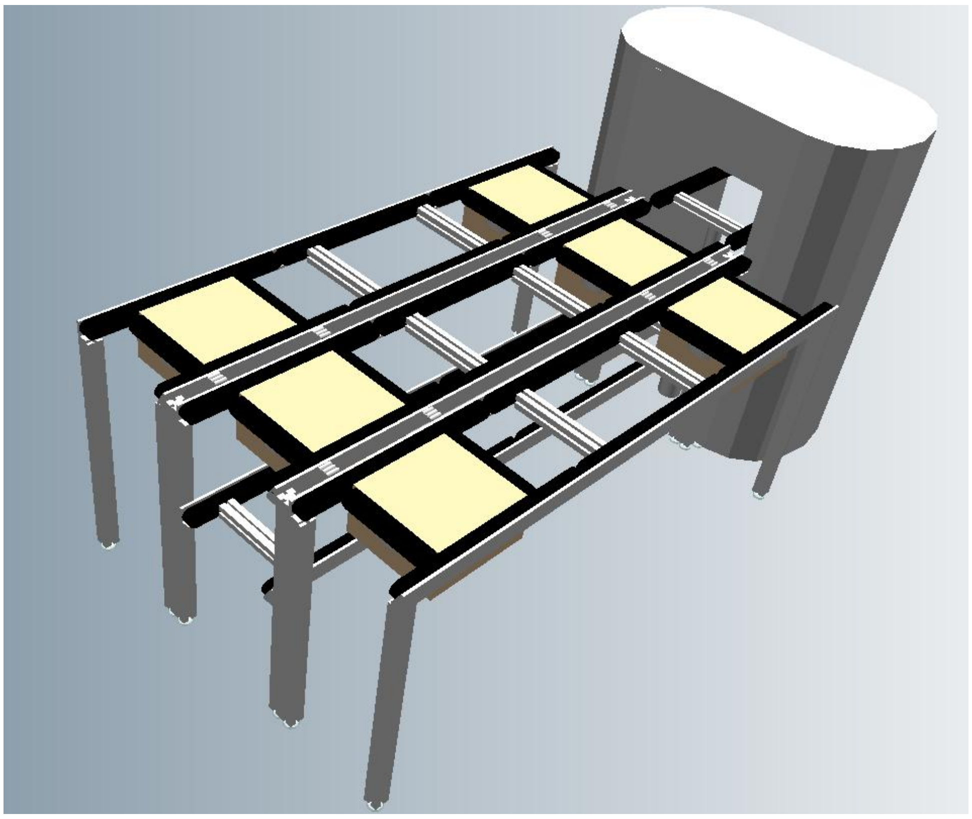


Figure 17 FESTO Studio model, middle segment

Objects that involve with the middle segment are:

- MiddleSegment.mod
    a) IntermediateLifter
    b) M_MainLine (as a portion of mainline, with two level of conveyors)
    c) M_RightLine (taking the pallet flowing direction as reference direction)
    d) M_LeftLine
- M_SimulationController

What is worth to mention here is that all cross conveyors of the first two segments are all implemented with a small push device, pushing up the conveyor when it needs to deliver pallet in the crossover direction. Normally, the cross conveyor is equipped a little lower than the main direction surface (see Figure 18). When crossover direction is selected by user, the pallet will stop on top of the cross conveyor, waiting for it to rise up and then roll on.

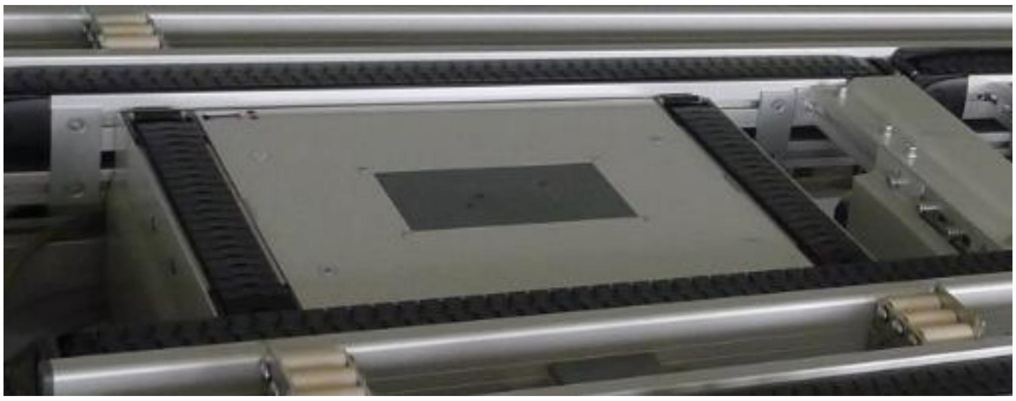


Figure 18 Cross conveyor

3.1.3. End segment

End segment (DAS Ergo) consists of an ergonomic manual workstation, an end lifter (5047EN-1HC) and a portion of the mainline.

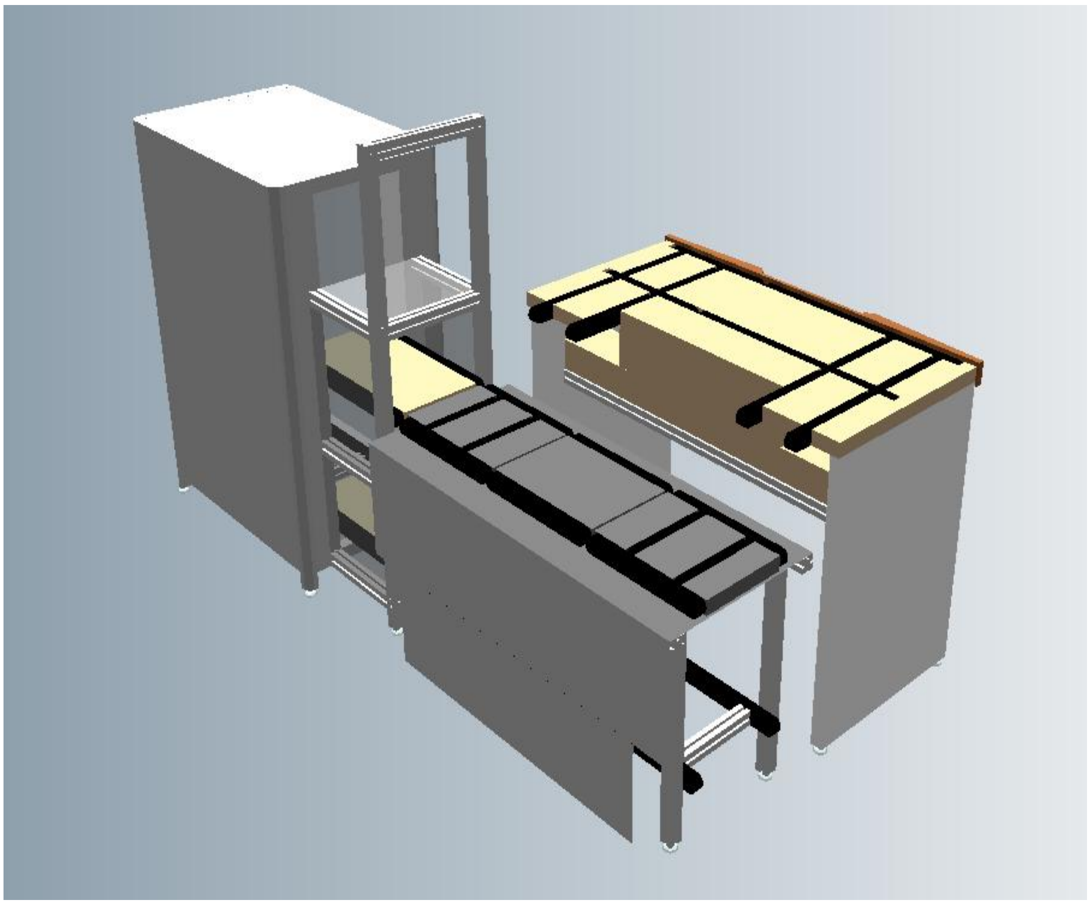


Figure 19 FESTO Studio model, end segment

Objects that involve with the end segment are:

- EndSegment.mod
    a) E_MainLine (as a portion of mainline, with two level of conveyors)

b) E_WorkStation
c) EndLifter

- E_SimulationController

3.2. Routing description

The pallet flowing logic in start segment is defined as depicted in Figure 20.

There are in total two decision points in start segment.

Firstly, when the pallet leaves the start lifter and fully occupies the first cell of mainline (s_ml2_c1), it needs to be told by the control system that which way it continues: mainline or workstation? If the pallet is chosen to go to the workstation path, then the pallet turns right (taking the mainline flowing direction as reference), and the route covers two side lines and the manual workstation, no other intervening needed; and if the pallet is chosen to go on the mainline route, it will reach to the second decision point (s_ml1_c2) after a short while, which is the cell in front of robot cell. At this time, the pallet needs the instruction of going to robot cell or continuing on mainline. The simulated scenario of the robot execution here is for the pallet stopping two seconds at the end of the conveyor, then pallet moves back to the mainline, continues on to the middle segment.

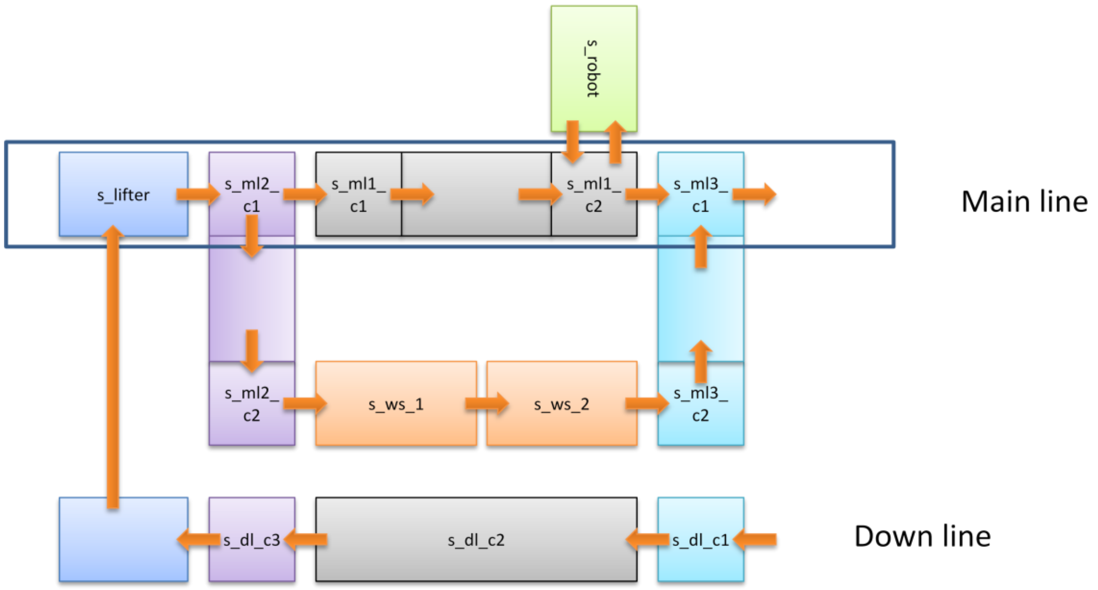


Figure 20 Start segment of conveyor system

The pallet flowing logic in middle segment is defined as depicted in Figure 21.

There are in total three decision points in middle segment, both on mainline.

The first cell (m_ml_c1) is equipped with cross conveyor and it requires direction decision from user: mainline or right line? However, no matter which route is determined, the pallet will ultimately reach the last cell of the mainline (m_ml1_c3) which is also the second decision point. At this point, the user has to make choice between continuing on mainline or going to the left line workstation. After all processes, the pallet will finally arrives at intermediate lifter, where it arrives at the third decision point where the pallet can either go down a little then pass to the end segment or down to the lower level conveyor as to return back to start segment.

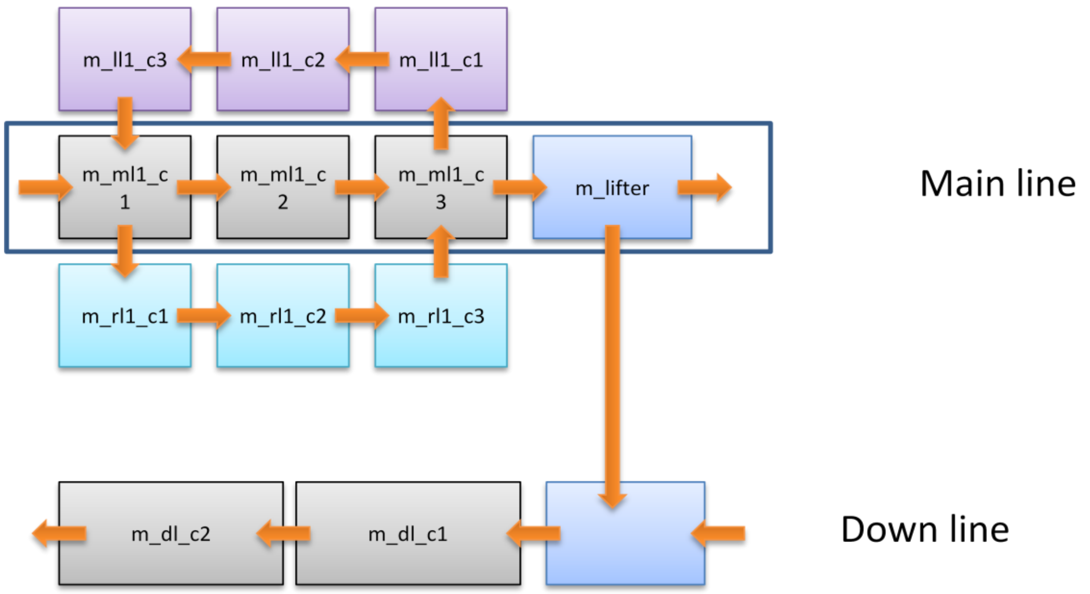


Figure 21 Middle segment of conveyor system

The pallet flowing logic in end segment is defined as depicted in Figure 22.

Only once in end segment does the decision point exist, which is the first cell (e_ml_c1) of mainline. It needs instruction to choose between mainline and workstation. If mainline rout is chosen, the pallet continues to the end lifter, translates down to the lower level and then returns back to the start. And if the workstation route is chosen, the pallet takes a detour to the workstation and then goes back to mainline.

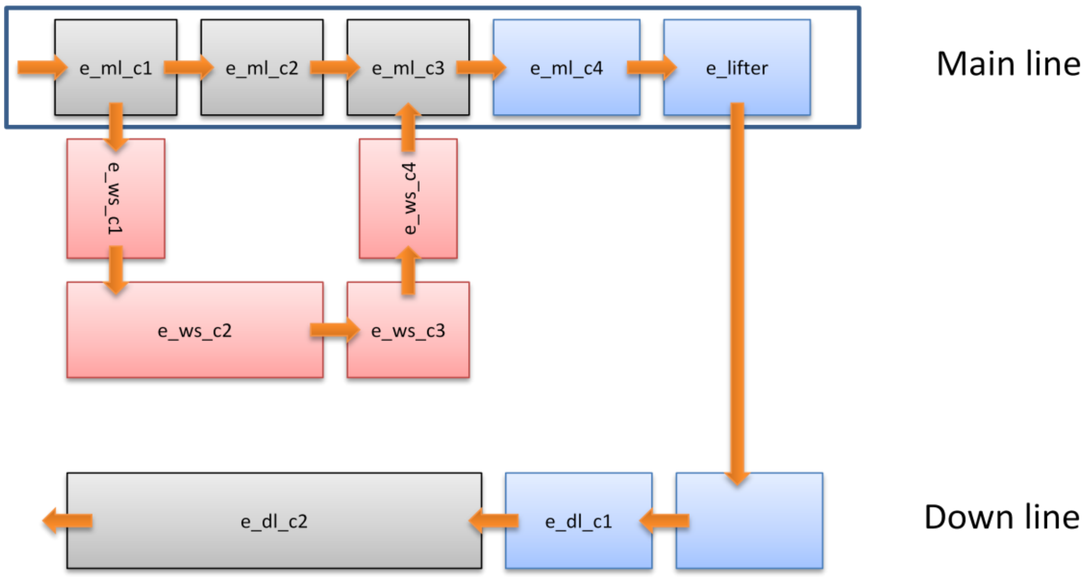


Figure 22 End segment of conveyor system

3.3. Mechanism application and I/O configuration

FESTO CIROS Studio software provides multiple libraries including a bunch of well-made robot models, controllers, miscellaneous mechanisms, sensors and modelling essentials. These models can be found under the menu of “Model libraries”. Meanwhile, users can always create their own model and save as a model for later reference.

In current production line, conveyors and lifters exist in every segment which requires motors to initiate physical translation movement and also sensors inserted in every conveyor. Therefore, certain miscellaneous mechanisms, such as ‘Conveyor belt’, ‘Reflex Light Barrier’ sensors, and ‘Cylinder for translation’, are widely utilized to simulate the functionality and performance of real actuators and sensors. How these mechanisms are implemented and how the inputs and outputs are configured in the model will be explained in this section.

3.3.1. Actuators

Mechanism ‘conveyor belt’ was used to simulate all the mechanical conveyors. This mechanism has two digital inputs ‘BeltOn’ and ‘BeltReverse’, which not only fulfils the functionality of a regular conveyor but also solve the problem that no motors can be modelled in the simulation. By setting and resetting digital inputs, conveyors could be turned on and off, move forward and backward.

Another application of actuator mechanism is using ‘Cylinder (two-way) for translation’ to complete the specifications of all lifters (allocated in start lifter, intermediate lifter, end lifter, and every cross conveyor). The lifter is composed of two parts, ‘Base’ and ‘Piston’. By inserting lifter frame (the body of lifter) into the ‘Piston’ section and defining the coordination of ‘Gripper point’, this translation model could imitate the movement of a real lifter as to move in and out smoothly. By changing the lower and upper axis limit (which can be found in ‘General’ tab from ‘Properties for section’ menu), the range of movement can be edited. For the purpose of carrying a pallet on the lifter, a commensurate ‘Grip point’ needs to be added on the pallet, where the pallet is grasped by the lifter. Thus, when the pallet moves onto the lifter and the ‘Grip point’ meets with ‘Gripper point’, grasping functionality is completed to enable the pallet to move with lifter all together.

A standard ‘Cylinder (two-way) for translation’ has two inputs, ‘MoveOut’ and ‘MoveIn’, and two outputs, ‘IsMovedOut’ and ‘IsMovedIn’. Inputs control the movement of translation and outputs indicate the status of movement. What is worth to notice here is that two inputs cannot be set at the same time; otherwise it may cause some confusion and the latter changed signal does not influence anything. Therefore, when programming the control logic, it is always always important to reset those controlling I/Os to their initial status. There is also another mechanism called ‘Cylinder for translation’ which performs similarly as the two-way cylinder, which has only one input ‘MoveOut’ and one output ‘IsMovedOut’. The difference between these two translations is that, during translation movement, two-way cylinder can be forced to change translation direction if both input signals are changed; however, translation direction of one-way cylinder can be changed only when the ‘Piston’ is moving in, which means once the signal of ‘MoveOut’ is changed from 0 to 1, the ‘Piston’ section could only move in after it reaches the upper axis limit.

Table 6 I/O descriptions of actuator mechanism

	Conveyor belt		**Cylinder (two-way) for translation**	
	I/O name	**Description**	**I/O name**	**Description**
Inputs	BeltOn	The conveyor is turned on, any object with a valid ‘Grip point’ can move on the conveyor surface by its default direction.	MoveOut	The ‘Piston’ section moves away from the ‘Base’ section.
	Bel-tReverse	This input must work associated with input ‘BeltOn’. With it set to be 1, the direction of conveying reverses.	MoveIn	The ‘Piston’ section moves back towards the ‘Base’ section.
Outputs	PartAtEnd	Report when the object reaches the end of the	IsMove-dOut	The ‘Piston’ section has moved away from the ‘Base’ section and reached

		conveyor surface.		its maximum limit.
			IsMov-edIn	The ‘Piston’ section has moved back into zero position referring to the ‘Base’ section and reached its minimum limit.
Attach-ments	Base	With a default ‘conveyor surface’ built in, of which the dimension and pose can be edited.	Base	With a default ‘Grip point’ built in.
			Piston	With a set of default ‘Gripper point’ built in.

Generally, there are two kinds of conveyors utilized in current model: cross conveyor and one-way conveyor. The cells that are facilities with cross conveyors offer two possibilities of directions. And in these cells, two sensors are embedded to determine whether the cell is fully occupied by the pallet, when to stop for instruction and when to move forward. Each cross conveyor is equipped with a small lifter for the purpose of transfer pallet to the other direction. When the non-main route is chosen, the lifter lifts up and creates face connections between pallet and conveyor belt, then the conveyor is ought to be turned on and finishes the operation.

3.3.2. Sensors

Mechanism ‘Reflex Light Barrier’, which provides ‘Detect’ and ‘Distance’ outputs, was implemented to simulate all the sensors in this model to recognize the existence of a blocking pallet. Based on customized demand, the measuring range of sensor could be edited in ‘Sensor’ tab from ‘Properties for object’. Behaviours of these sensors can be observed in ‘Manual Operation’ window as a light signal, which turns green if the sensor is occupied. In current model, only ‘Detect’ signal is used to locate pallet position and manage the conveyor movement as a control signal in later implementation phase.

Table 7 I/O descriptions for sensor mechanism

	Reflex Light Barrier	
	I/O name	Description
Outputs	Detect	Boolean variable, report when an object being discovered within its measuring range.
	Distance	Report the exact distance between the object within measuring range and the sensor using

		unit of millimetre.
Attachments	Base	With the default geometric shape of the sensor built in.

3.4. Design limitation and solutions

Current simulation model was built based on the actual outlook and measurements of the real Assembly line. But there are still some remaining limitations that cannot be fully expressed, and some of them that have been realized during simulation process are listed below.

- Positions of sensors inserted in the lifter

Observing the Assembly line, one can easily find that each cross conveyor/lifter is equipped with one or two sensors inserted. Normally, these sensors are meant to move up down with the lifter. However, during modelling it has been discovered that only simple objects (lifter frame, built from geometric primitives) but not existing models (sensors) can be inserted into ‘Piston’ section, so that sensors in the simulation cannot move with the lifter like the original conveyors but only stays where they were allocated initially.

Solution to this problem is to create a ‘Gripper point’ on the lifter and then attach the sensor on this ‘Gripper point’. To do this, the ‘Grip point’ created on the sensor needs to be grasped initially by lifter and also avoid locating on the track of nearby conveyor surface. Otherwise, when the nearby conveyor is turned on, the sensor maybe recognized as the object for transporting due to its ‘Grip point’.

- Duplication of conveyor surface

A standard physical conveyor possesses a conveyor surface on top of it, which is the interface of transporting objects, so is the conveyor mechanism offered in FESTO CIROS Studio. The size of conveyor surface constrains the range of conveying area. In reality, this conveyor surface moves with the conveyor simply because it is physically attached to conveyor despite of conveyor movement. However in FESTO CIROS Studio, when simulating lifters, a ready-made conveyor mechanism cannot be inserted into the ‘Piston’ section, which means it is not possible to model a movable conveyor. What is also worth to be noticed is that when placing conveyor surface, two surfaces from near conveyor segment should not overlay on each other. Otherwise, when the first conveyor stops, even though the second is still on, pallet may be observed to stop on a “running” conveyor.

Solution here is to duplicate the conveyor surface for the other layer of conveyor and only keep one lifter frame. For example, the lower layer of start lifter is a perfectly functioned conveyor; when it moves out towards the upper layer, only the 'Piston' section (lifter frame) raises until it reaches fully move out position and meets with the upper layer conveyor surface. This upper layer "conveyor" is made of only an invisible conveyor surface in the simulation model which has its own control I/Os. For this case, each lifter is allocated with two sets of control signals dedicated to upper and lower layer separately.

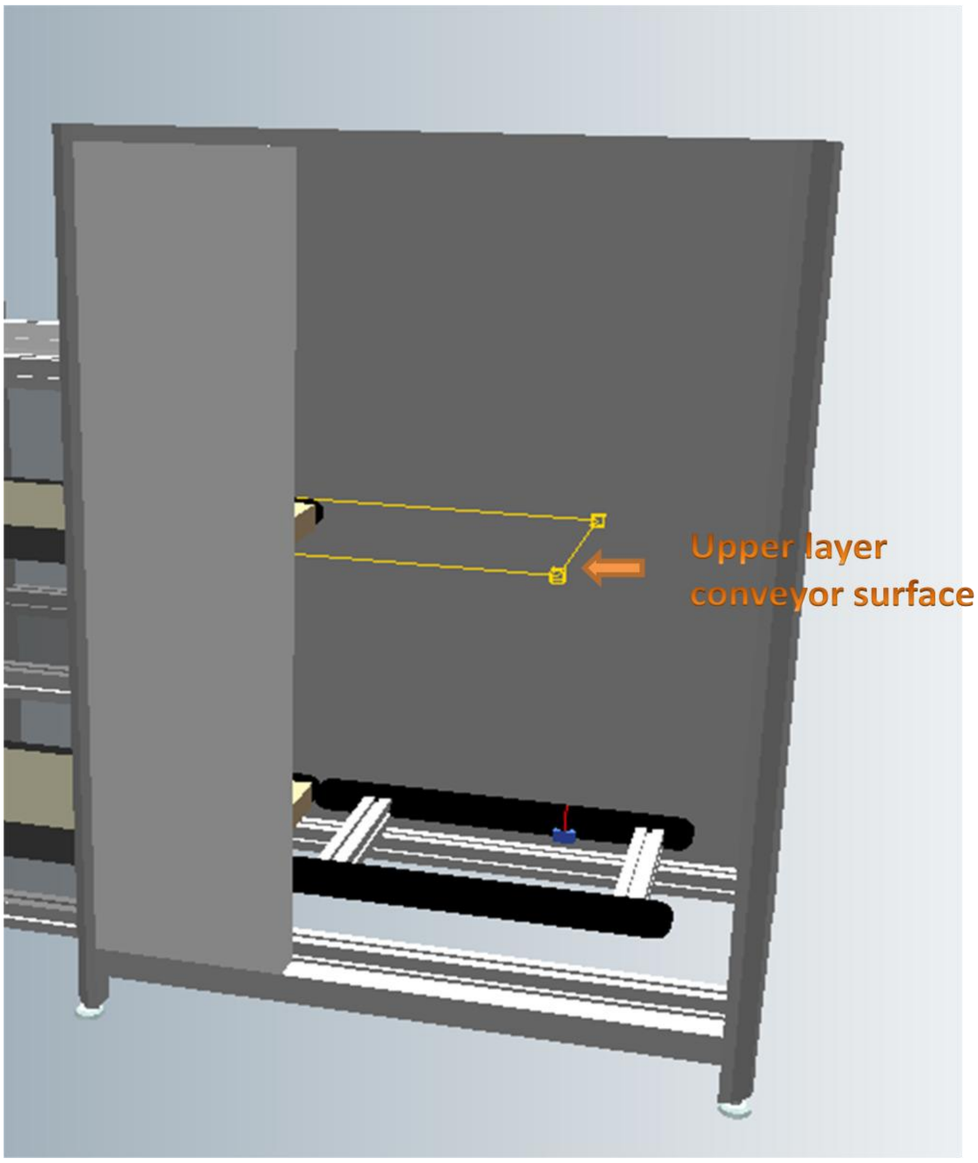


Figure 23 End lifter conveyor surface

- Sensor does not work when 'Distance' is measured to be 0

During the simulation process, it has been discovered that the sensors stop working when the surface of the object, which is supposed to be detected, is sensed right upon the sensors. This circumstance happens when the output 'Distance' appears to be 0, even with an object existing within the measuring range of sensor, the output ‘Detect’ still turns to be 0. However, in reality it is always possible for a sensor to perform even no space remains between the sensor and the surface upon it.

Solution to this problem is simply locating every sensor a bit shallower than the surface of embedded object, for instance 10mm.

- Lifter is not stoppable during translation

For safety considerations, any section out of the production line had better possess an emergency stop control, including all the lifters. If any abnormality occurs, it should be possible for the operator or inspector to stop the lifter from moving manually. This requires the mechanism for simulating lifter to offer a ‘stop’ input to terminate the translation movement. However, neither of the two mechanisms (regular and two-way cylinder for translation) mentioned previously provides a separate input for this purpose. After all, the current model was built under the condition that no other human intervention is needed during the whole process.

This limitation also causes another problem in the simulation, which is multiple stop during lifter translation. Out of all three lifters in the production line, the intermediate lifter is distinguished from the other two due to its multiple layers stop. The height of mainline before middle segment (including middle segment) is 930mm, and 815mm for the mainline of end segment; the height of down line before middle segment is 480mm, and 255mm for the down line of end segment. These height differences requires the intermediate lifter to have four stops (see Figure 24) during translation because it responses for delivering the pallet on either mainline or down line from middle to end segment, vice versa. Besides, it should also be able to execute the possibility to transfer pallet from upper layer to lower layer than back to start.

The most complex lifter is the intermediate lifter, which especially requires the feature of sudden stop during translation. What was utilized to simulate this special lifter is a conveyor mechanism modelled as the vertical track for lifter movement. Inserting a ‘Grip point’ on the lifter frame, the lifter can move as a “pallet” on its active conveyor surface. At the same time, four sensors are employed on each level of stop in order to control the behaviour of lifter, when and where to stop. Integrated with these four sensors, the timing for lifter to stop is controlled by the combination of sensor output and conveyor input.

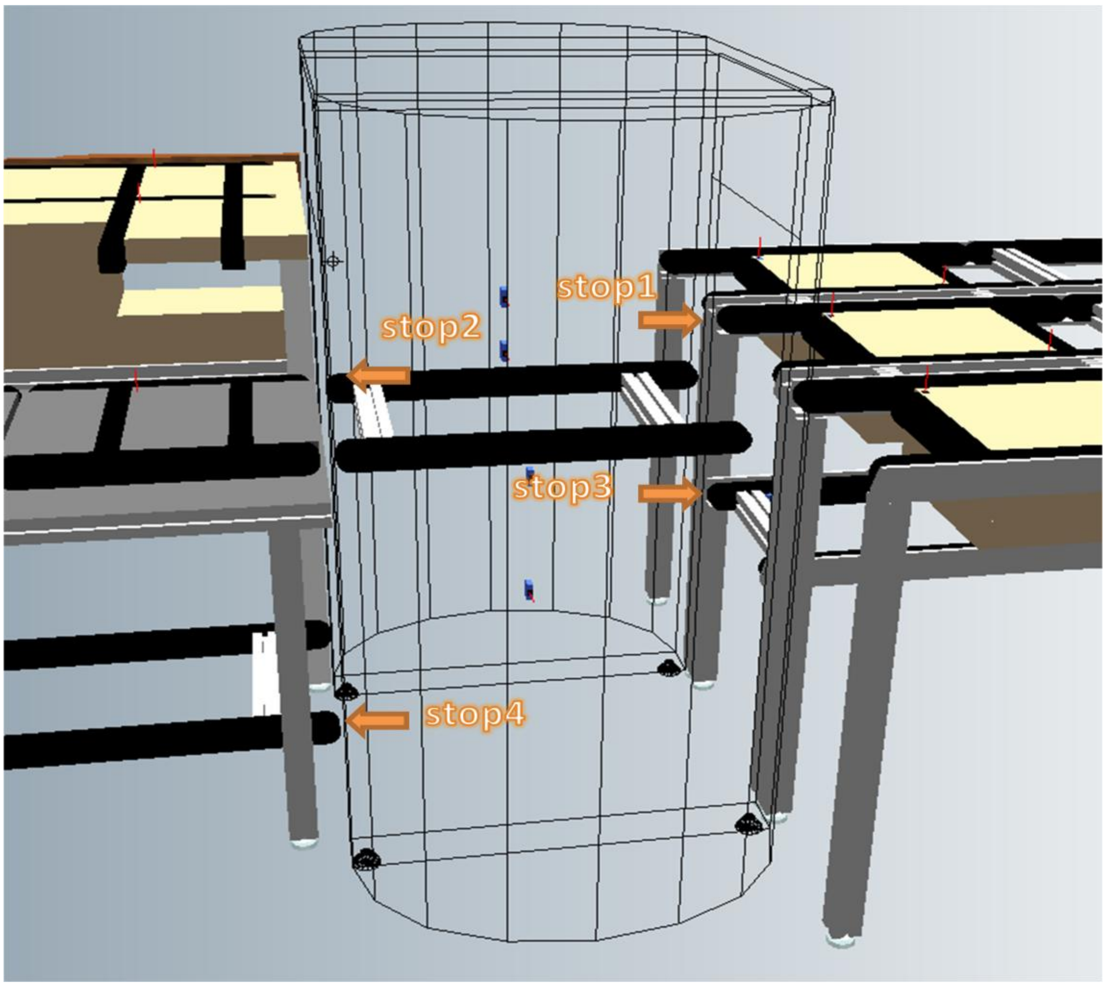


Figure 24 Intermediate lifter

- Laser line of sensor is not removable

As one can see in the simulation model, there are various noticeable little red lines emitted from the surface of conveyors. These are the simulated laser line of sensor. As mentioned previously, the measuring range of the sensor can be edited and it can be identified as the length of red laser line. The reason that this has been brought up as a design drawback is out of aesthetic consideration. The model would appear more likely to the real production line if the laser line can be set invisible like any other objects.

- Pallet stops on cross conveyor

Out of the whole production line, there are plenty of perpendicular corner where the pallet needs to turn 90 degree on the conveyor surface to continue moving on another line, and these are so called cross conveyors. When pallet reaches these cross conveyors, it either continue on the same direction that it came from or change for a perpendicular direction, for example, cell m_ml_c1.

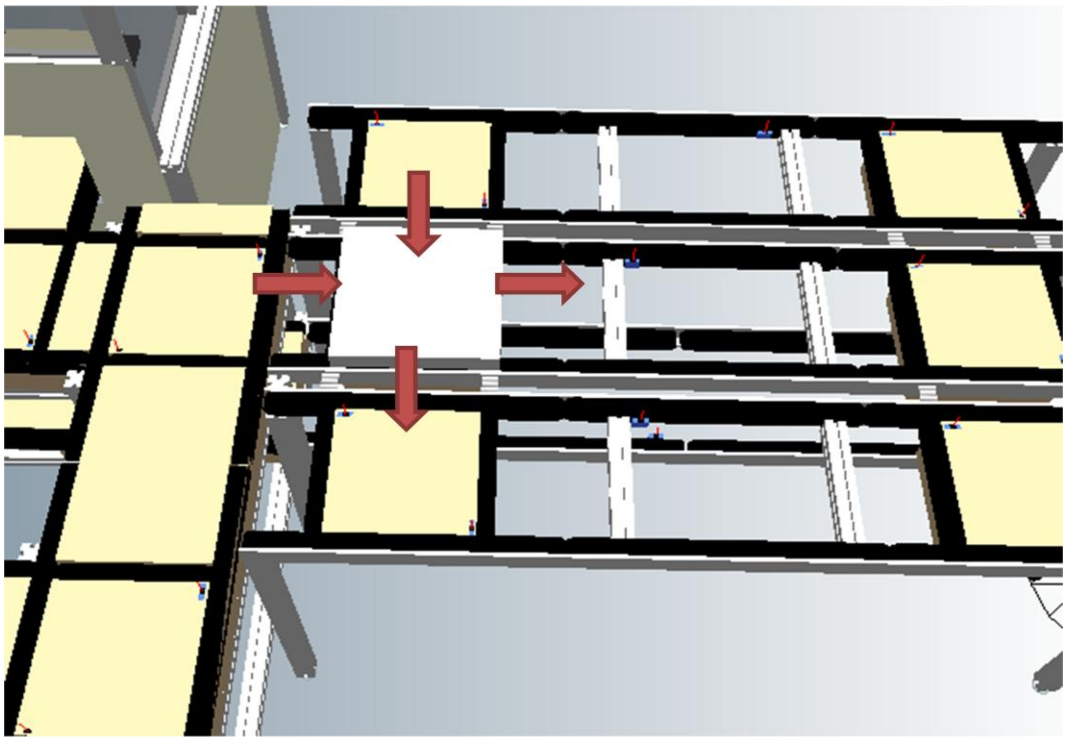


Figure 25 Cell m_ml_c1

There are two input possibilities for this cell, from start segment or the left line, and two output possibilities, to mainline or the right line direction. Decision needs to be determined when pallet stops in the position shown in Figure 25 and the decision is involved as the form of an input signal. If the pallet is loaded from start segment and decision says 'continue on mainline', then there is no obstacle for the pallet to move on. However, if pallet is loaded from start segment and decision says ' go to the right line', somehow it was discovered during the simulation test that the pallet does not move even though the cross conveyor is turned on (can be observed from 'Manual operation' window). The same situation encounters when the pallet is loaded from left line and meant to move on the mainline next. Once this problem occurs, it needs a little 'push' to pave the way of following movement, which is done by edit the (x, y, z) parameters in 'Pose' tab.

The solution to this problem has been discovered during experiments. If a pallet is determined to make a 90 degree turn on two perpendicular conveyor surfaces, the essential condition is that the pallet must reaches the end of first conveyor then it can automatically change for another condition. The 'PartAtEnd' output of conveyor mechanism can be utilized to acknowledge when a pallet is unloaded from current conveyor. So to build rather shorter conveyor surface before cross conveyor is the solution for many corner cell in the line, such as s_ml2_c2, m_rl_c1, and m_ll_c1 and so on.

When applying this method on the cells with two inlets and two outlets (three cells in total, which are s_robot, m_ml_c1 and m_ml_c3), a conflict between production reality and simulation smoothing encountered.

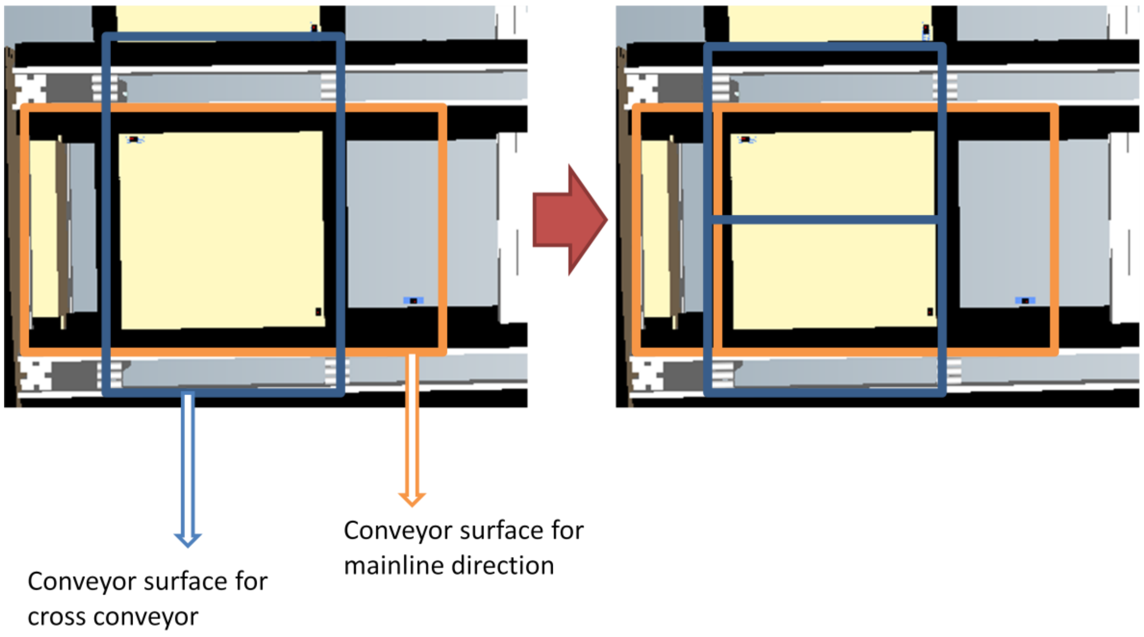

Conveyor surface for mainline direction
Conveyor surface for cross conveyor

Figure 26 Solution for cells with two inlets and two outlets

As shown in Figure 26, the orange square represents the conveyor surface for mainline direction and the blue one represents the conveyor surface for cross conveyor. Normally the configuration of conveyor surface should be looking like the left picture, which is one conveyor for each direction. Due to one control signal maps to one conveyor, there is only one control signal responding to one direction, which is also the reality in actual production line. However, in simulation, this configuration is not qualified for the demand of smooth transportation as pallet always stopping on cross conveyor even though the conveyor appears to be turned on. For the purpose of continuous simulation, conveyor from each direction is divided into two parts (right picture). In this way when the pallet passes through the first part, it reaches the end of the conveyor surface, thus it can continue move on any other direction according to the decision program.

4. SIMULATION IMPLEMENTATION

With the 3D simulation model of the assembly line ready, integration of PLCs was next employed into the model. In FESTO CIROS Studio, there are several simulation controllers existing in model library, out of which S7-simulator is the one that interprets executable S7-Programs.

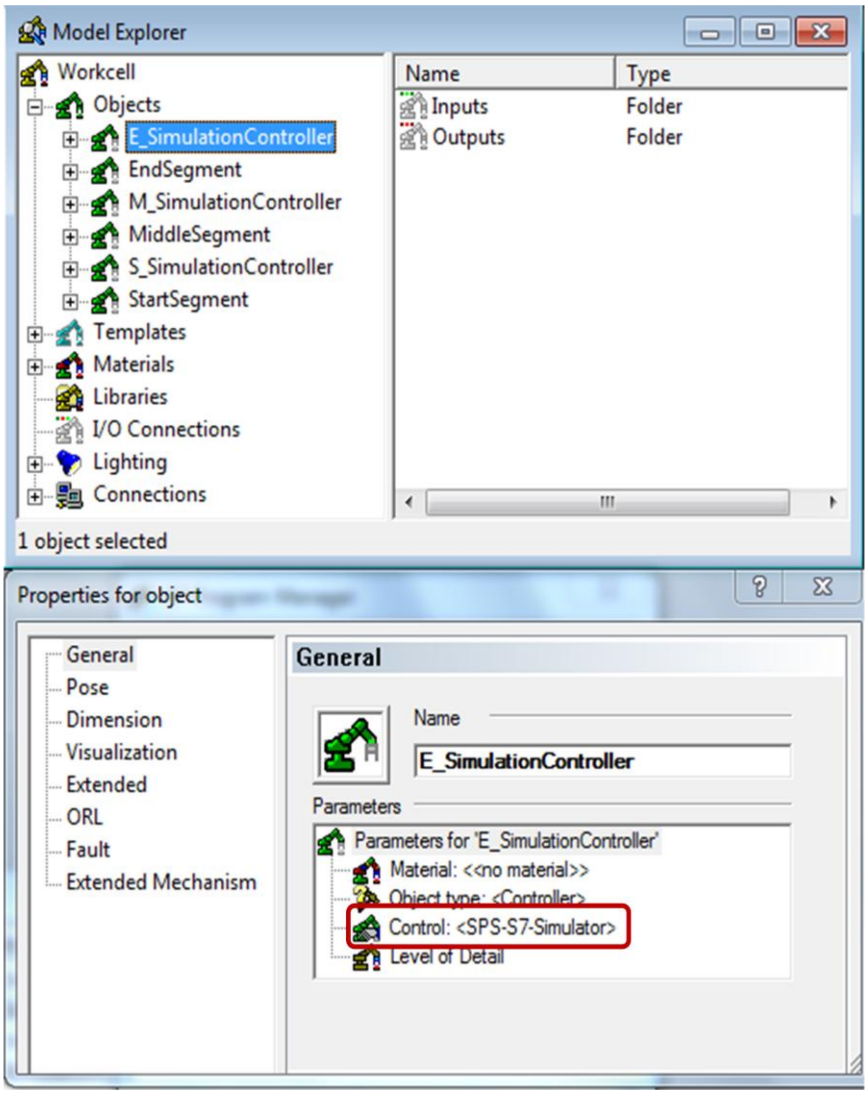
The SIMATIC manager is used for programming PLC. This software is part of the PLC development environment STEP 7 distributed by the Siemens cooperation. The whole configuration of a PLC is stored in a S7 project. The language that used in programming PLC in current thesis is FBD and STL.

4.1. Implementation steps

There are three steps to integrate a PLC into the current model:

4.1.1. Insert the PLC into the model

After establishing the simulation model of the assembly line, three simulation controllers were inserted into the model, each of which controls one segment of the line. One can go to the global properties of the object to change the type to controller from ‘Time’ to ‘SPS-S7-Simulator’. By this time, the simulation model should be able to perform some functions by pressing ‘Simulate’ button and changing the signals of the inputs in ‘Manual Operation’. And it is recommend by the author to test all the mechanisms in this phase, for instance, the size of a conveyor surface or the maximum axis limit of a translation cylinder.


Model Explorer

Workcell
- Objects
  - E_SimulationController
  - EndSegment
  - M_SimulationController
  - MiddleSegment
  - S_SimulationController
  - StartSegment
- Templates
- Materials
- Libraries
- I/O Connections
- Lighting
- Connections

Name	Type
Inputs	Folder
Outputs	Folder

1 object selected

Properties for object

General
Pose
Dimension
Visualization
Extended
ORL
Fault
Extended Mechanism

General

Name: E_SimulationController

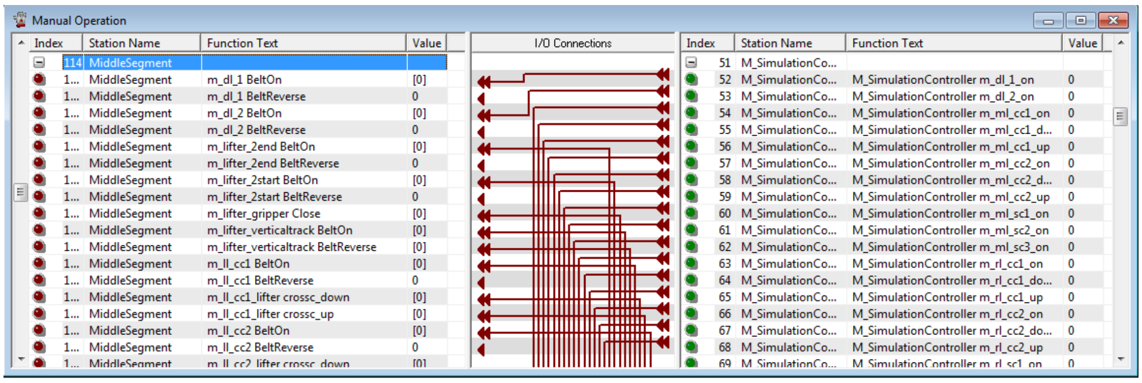
Parameters

Parameters for 'E_SimulationController'
- Material: <<no material>>
- Object type: <Controller>
- Control: <SPS-S7-Simulator>
- Level of Detail

Figure 27 Simulation controllers setting

4.1.2. Link the I/Os of the PLC with the objects in the model

When all the mechanisms are verified to be working fine manually, the connections between actuators, sensors and PLC is ought to be built next. One can add new digital inputs and outputs to the controller by right clicking on the controller name. Then these unnamed I/Os will be listing in ‘Manual Operation’ window. By giving them dedicated names, it is easy to recognize the functionality of each I/O. Then one must link the I/O of controllers to the simulated objects, controller inputs to object outputs and controller outputs to object inputs.



Manual Operation

Index	Station Name	Function Text	Value
114	MiddleSegment		
1...	MiddleSegment	m_dl_1 BeltOn	[0]
1...	MiddleSegment	m_dl_1 BeltReverse	0
1...	MiddleSegment	m_dl_2 BeltOn	[0]
1...	MiddleSegment	m_dl_2 BeltReverse	0
1...	MiddleSegment	m_lifter_2end BeltOn	[0]
1...	MiddleSegment	m_lifter_2end BeltReverse	0
1...	MiddleSegment	m_lifter_2start BeltOn	[0]
1...	MiddleSegment	m_lifter_2start BeltReverse	0
1...	MiddleSegment	m_lifter_gripper Close	[0]
1...	MiddleSegment	m_lifter_verticaltrack BeltOn	[0]
1...	MiddleSegment	m_lifter_verticaltrack BeltReverse	[0]
1...	MiddleSegment	m_ll_cc1 BeltOn	[0]
1...	MiddleSegment	m_ll_cc1 BeltReverse	0
1...	MiddleSegment	m_ll_cc1_lifter crossc_down	[0]
1...	MiddleSegment	m_ll_cc1_lifter crossc_up	[0]
1...	MiddleSegment	m_ll_cc2 BeltOn	[0]
1...	MiddleSegment	m_ll_cc2 BeltReverse	0
1...	MiddleSegment	m_ll_cc2_lifter crossc_down	[0]

I/O Connections

Index	Station Name	Function Text	Value
51	M_SimulationCo...		
52	M_SimulationCo...	M_SimulationController m_dl_1_on	0
53	M_SimulationCo...	M_SimulationController m_dl_2_on	0
54	M_SimulationCo...	M_SimulationController m_ml_cc1_on	0
55	M_SimulationCo...	M_SimulationController m_ml_cc1_d...	0
56	M_SimulationCo...	M_SimulationController m_ml_cc1_up	0
57	M_SimulationCo...	M_SimulationController m_ml_cc2_on	0
58	M_SimulationCo...	M_SimulationController m_ml_cc2_d...	0
59	M_SimulationCo...	M_SimulationController m_ml_cc2_up	0
60	M_SimulationCo...	M_SimulationController m_ml_sc1_on	0
61	M_SimulationCo...	M_SimulationController m_ml_sc2_on	0
62	M_SimulationCo...	M_SimulationController m_ml_sc3_on	0
63	M_SimulationCo...	M_SimulationController m_rl_cc1_on	0
64	M_SimulationCo...	M_SimulationController m_rl_cc1_do...	0
65	M_SimulationCo...	M_SimulationController m_rl_cc1_up	0
66	M_SimulationCo...	M_SimulationController m_rl_cc2_on	0
67	M_SimulationCo...	M_SimulationController m_rl_cc2_do...	0
68	M_SimulationCo...	M_SimulationController m_rl_cc2_up	0
69	M_SimulationCo...	M_SimulationController m_rl_sc1_on	0

Figure 28 Manual Operation in FESTO CIROS Studio, I/O connection

4.1.3. PLC programming

The programming of PLC is done by SIMATIC manager. Each S7 project could include several S7 programs separately. Before programming, it is very important to define all I/Os of the controller and assign them according to the order listed in simulation controller. This can be done by clicking on ‘Symbols’ section after creating a new program in S7 project. Also each control signal should have its own unique name regarding to its functionality. Then one is ready to start programming PLCs in ‘Block’ section. Function Block Diagram and Statement List are used in current program and some example is illustrated in Figure 29.

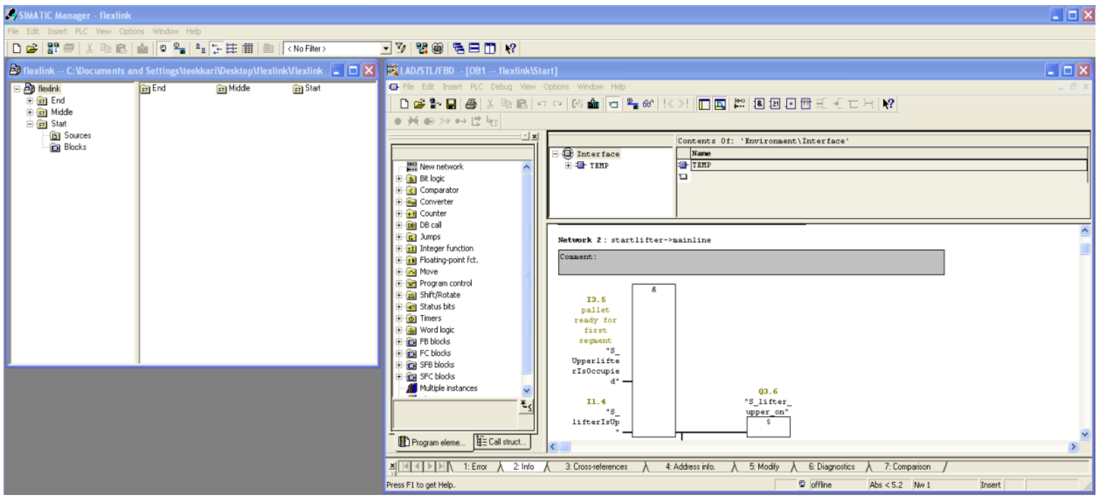


Figure 29 SIMATIC manager

4.2. Design specifications

During program design phase, plenty of production features need to be noticed closely. Some features can be only realized by means of programming, and these issues are summarized in the following sections.

4.2.1. Timing for pallet stopping

Among all the cross conveyors, there are some that the pallet needs to stop on either waiting for instruction of next move or turn perpendicularly. All these cross conveyors are equipped with two sensors, one for regular direction control and the other for crossing direction control. The outputs of these sensors help to determine when and where the pallet should stop. And the pallet is ought to stop when it covers right above the cross conveyor, including both sensors. In the PLC program, a Start On-Delay Timer is employed to compute the timing for pallet stopping and the control logic is: when both sensors of a cross conveyor report to detect and the conveyor of current direction are on, after little amount of time (350-600ms was used in the program), turn off the conveyor.

4.2.2. Interactions between segments

Due to the fact that three controllers are utilized in the model instead of one, some communication needs to be established between two nearby segments. When pallet moves between two segments, it is mandatory for both segments to set up some connections to communicate the status of pallet flowing. One of the importances is that the latter segment should be able to notice former segment and shut down some unnecessary conveyors. Therefore, some signals from latter segment should also be considered when making logic decisions for former segment. For example, in Figure 30, the first two sensors (marked in red circle) from middle segment are extended in the logic control for start segment. The connection can be built up in 'Manual Operation' window, and it is worth to notice that only outputs from the model can be connected to multiple controller inputs. It is not possible for inputs to have multiple control outputs manipulating at the same time.

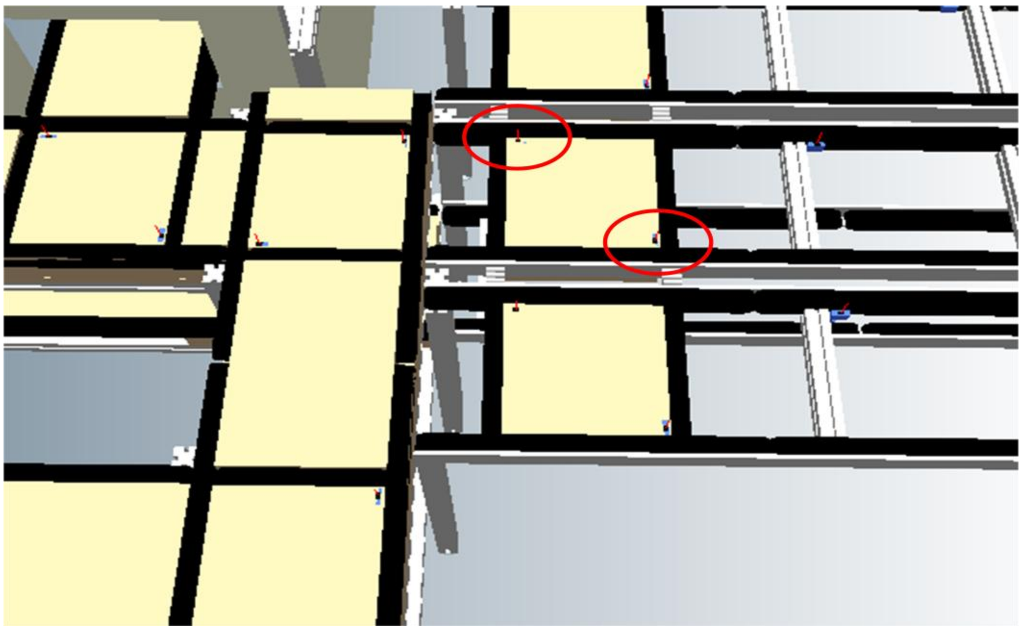


Figure 30 Interactions between segments

Due to the complexity of the intermediate lifter, that it is the one and only connection section between middle and end segment for both upper and lower layer, sensor signals from both layers has to be involved in the PLC for middle segment. The main functions that the intermediate lifter features are:

1. Pick the pallet from upper layer, middle segment, grasp it and transfer it to upper layer, end segment as well
2. Pick the pallet from upper layer, middle segment, grasp it and transfer it to lower layer, middle segment
3. Pick the pallet from lower layer, end segment, grasp it and transfer it to lower layer, middle segment

Three status signals from end segment are considered in the simulation controller for middle segment. Two are from upper layer, which are the first sensor embedded in cell e_ml_c1 and the 'IsMovedIn' output signal from the lifter in the same cell reporting when the lifter is down to the upper layer surface. The sensor signal involves in the control logic for stopping the upper layer conveyor of the intermediate lifter (815mm in height). And the 'IsMovedIn' signal, which implies when the lifter is in default position, is one of the conditions that the pallet starts leaving intermediate lifter and moving towards end segment. For the lower layer, the sensor inserted in the lower layer of end segment mainline reports to middle segment PLC, which calls for the intermediate lifter to pick up pallet when it detects one loading on the lower layer. The pallet will stop upon the sensor until the intermediate lifter reaches its position. Since there are two cells which are qualified to call for the intermediate lifter for picking (cell m_ml_c3 and e_dl_c2), in case that one cell is calling when the intermediate lifter is processing on one of the other levels, one mandatory condition for moving intermediate lifter is that none of the rest levels conveyors is on.

4.2.3. Multiple pallets handling

When operating the actual production line, processing multiple pallets at the same time is a crucial property for any production procedure. This brings efficiency in both time and financial aspect. The capability of handling multiple pallets in the simulation is realized by programming PLC to coordinate among nearby cells.

There are two phases regarding to the design of cell coordination. The core concept of phase one is to shut down the conveyor which just unloaded the pallet. Thus, if the sensor from current cell detects existence of pallet and the current conveyor is on, the conveyor from previous cell should be shut down. In this way, after a single pallet flows over the whole production line (on both layers), all the conveyors are set back to default status which is off.

The second design phase is to extend the time duration of shutting down the previous conveyor only if there is no pallet following up, which should be seen from the actual production processes; and also ask the next cell if it is occupied before loading the pallet to it. The main principle is, when the pallet is detected by the sensors of current cell and meanwhile the sensors from previous cell have no response, which means no pallet is in the coming cell, it is proper time to shut down the previous conveyor; at the same time, if the sensors from next cell are acknowledged to be busy, stop the current conveyor until the next cell is idle. In order to realize this, a status variable is introduced in the program which regards a cell as a unit and this variable indicates the working status of the corresponding cell. When the status variable is set to be 1, it means that the cell is busy processing. And by busy processing, it means that both sensors (in some case there maybe only one sensor) are occupied, and by idle, it means neither or the sensors is occupied. So in simple words, the principle can be translated into: when the current cell is busy processing, if the previous cell is busy processing (with a new pallet), remain the current configuration; if the previous cell is not busy processing, turn off its conveyor; when the current cell is ready to unload the pallet to next cell, if the next cell is busy, stop the current conveyor; if the next cell is idle, turn the current conveyor on and load the pallet to the next cell.

4.2.4. Decision signals

There are totally six decision points in the whole assembly line, out of which two are from start segment, three are from middle segment and one is from end segment. They are summarized as shown in the following table.

Table 8 Decision points and corresponding inputs

Segment name	Cell name	Direction options	Input name
Start segment	s_ml2_c1	mainline direction/ workstation direction	S1_mainline S1_workstation
	s_ml1_c2	mainline direction/ robot cell	S2_mainline S2_robotcell
Middle segment	m_ml_c1	mainline direction/ right line	M1_mainline M1_rightline
	m_ml_c3	mainline direction/ left line	M2_mainline M2_leftline
	m_intermediatelifter	end segment direction/ start segment direction	M3_continue M3_return

End segment	e_ml_c1	mainline direction/ workstation direction	E_mainline E_workstation

4.3. User Interface and recipe loading

On the side face of the start lifter, there is a user interface built with four groups of buttons which can be seen in Figure 31. These interfaces are only designed for user's benefit of easy access to recipe determination, and they are not included in the original assembly line.

The orange button on the very top is the button for sending out new pallets. Each press of the button corresponds to generate a new pallet on the lower layer of start lifter. This function is realized by inserting a replicator mechanism into the simulation model. The replicator mechanism provides the ability to generate new objects based on templates. By creating a pallet-shaped template and setting the input of replicator from 0 to 1, a new pallet with extended name is generated from its first grip point position. In current model, the start point is set to be (-830, 7.5, 530). When pressing the button, a new pallet with a new name is generated at start point.

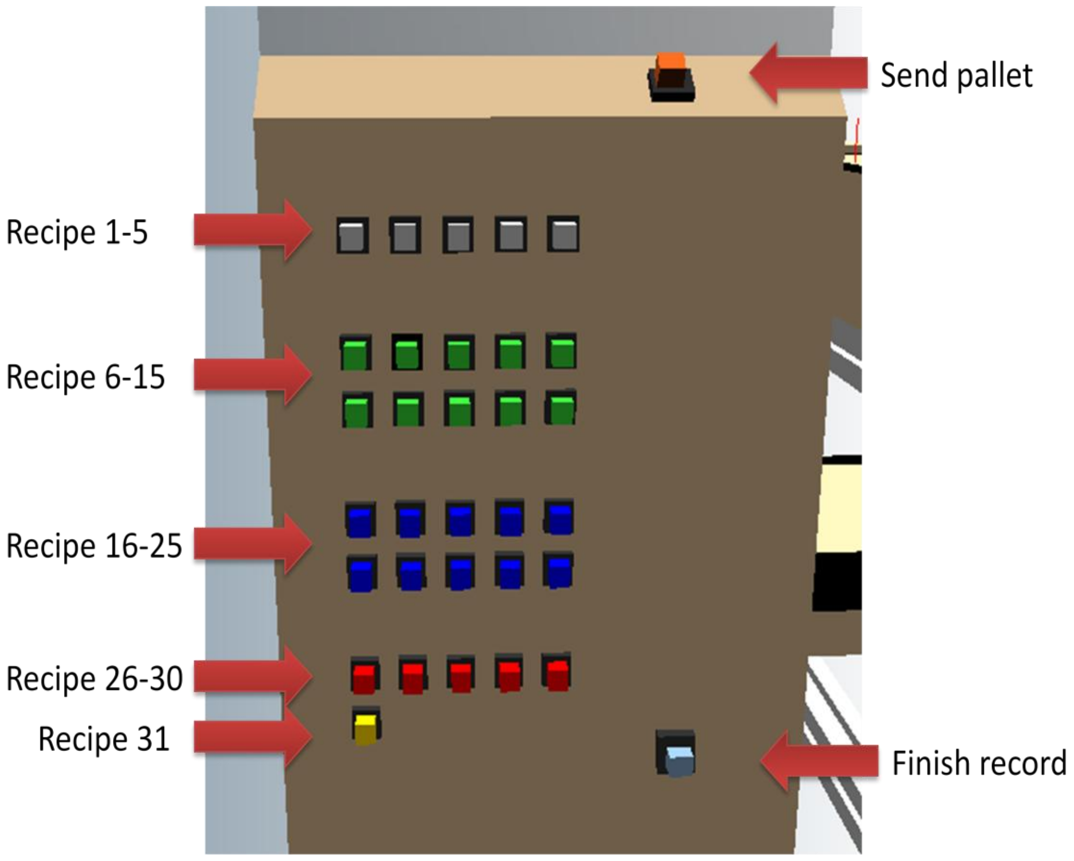


Figure 31 User interface for recipes

There are five working cells in the whole line: robotcell, workstation from start segment, rightline and leftline from middle segment and workstation from end segment. By sorting these five stations in different combinations, the recipes are defined correspondingly. Thus 31 combinations map to 31 recipes, which can be summarized in the table of next chapter. Recipe 1-5 includes one working station; recipe 6-15 includes two stations; recipe 16-25 includes three stations; recipe 26-30 includes four stations and recipe 31 covers all five working stations. A controller was implemented dedicatedly for the determination of recipes, which is called “RecipeController” in the simulation model. This controller is connected with all the recipe buttons and all the decision signals. By pressing the recipe button, the corresponding decision signals are being stored which decides the route of the next one or next batch of pallets.

The grey button on the left down corner functions as an acknowledge signal which should be pressed when finishing entering recipes that need to be executed, informing the controller that it is ready for loading recipes.

In the program of recipe controller, 33 data blocks were built in advance to store recipes. Data block 1-31 stores the detail of 31 recipes correspondingly. Opening one data block, one can find an array with five elements on one dimension, representing five working stations. As shown in Figure 32, each cell is given a unique number, and the cells with red square are the cells where process stations locate, number 2(robotcell), 10(s_workstation), 15(rightline), 18(leftline) and 25(e_workstation). If the pallet needs the jobs carried out in certain stations, place the cell number in corresponding position; if not, set the element to be -1. For example, recipe 19 requires pallets to go to robotcell, rightline and leftline, the recipe is written as [2, -1, 15, 18, -1].

Having pre-defined all 31 recipes, the recipe data blocks are ready to use.

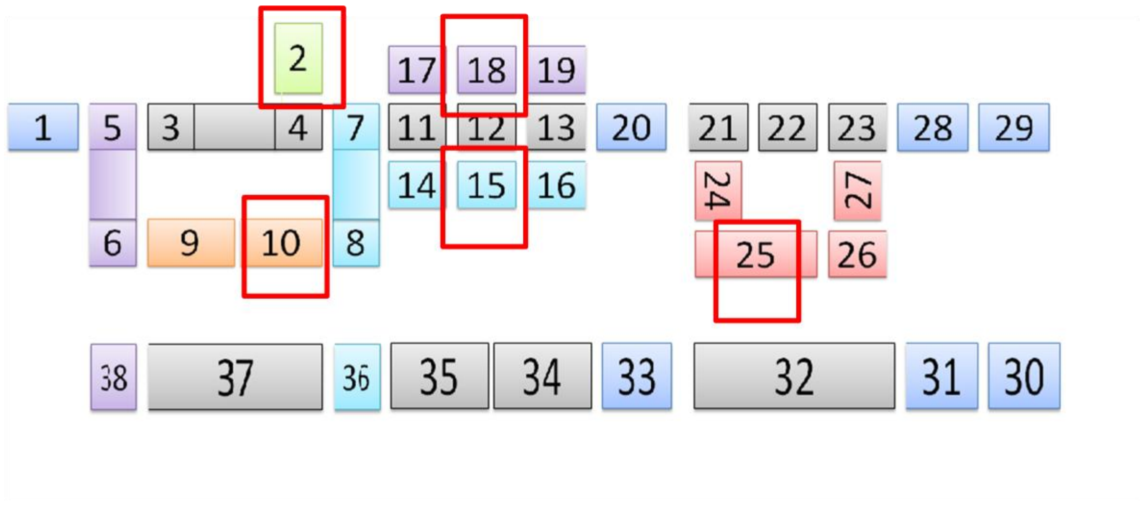


Figure 32 Cells numbering

Figure 33 gives an example of the process of loading recipes.

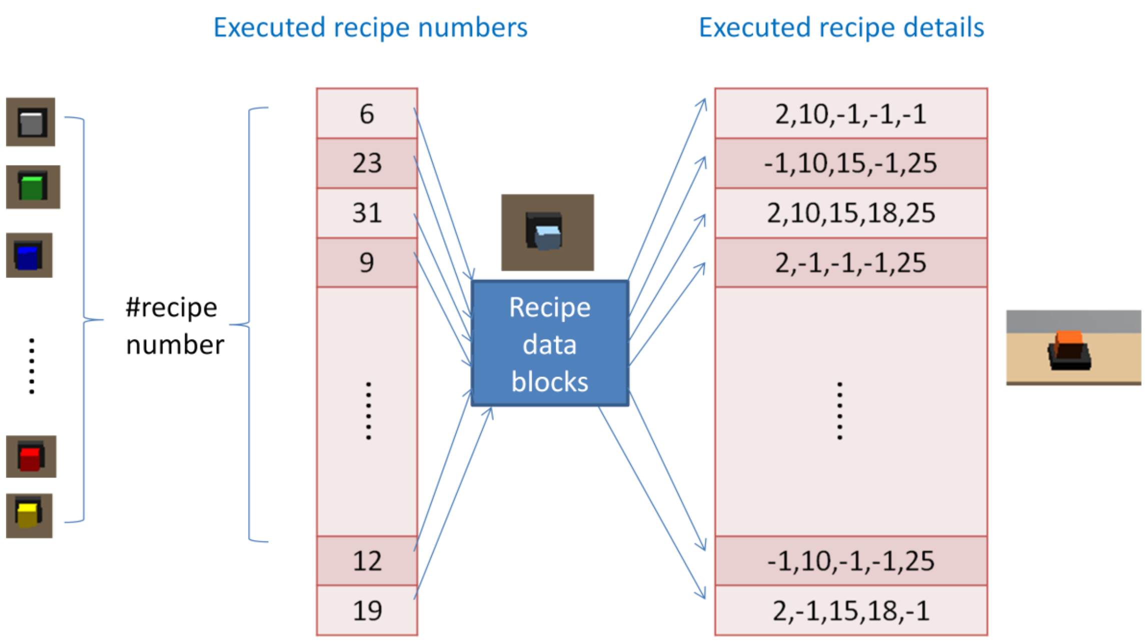


Figure 33 Recipes loading

When starting the simulation, what recipes are desired to be executed needs to be recorded firstly. By pressing the recipe buttons on side of the start segment, the corresponding recipe number is loaded to an integer type symbol out of memory area, #recipe number. Then an array with desired recipe numbers can be created by copying the content in #recipe number after each press. This array is stored in a new data block which is defined as an array with data length of 31 integers, which means at most 31 recipes can be recorded at once.

After recording recipe numbers, user must press on the grey button to acknowledge the termination of recording procedure, which does not only imply finishing creating the array of desired recipe numbers but also initiates the translation from recipe numbers to detail combinations of each recipe into the executed data block.

In the program, there are two customized functions for this phase, one (FC 1) for recording the pressed numbers and the other one (FC 2) for copying the corresponding recipe details from recipe data blocks to the memory area first and then pasting them to the executed data block.

4.4. Pallet tracking

As long as digging into the replicator mechanism offered by FESTO CIRSO Studio, it is discovered that there is no difference among the generated objects except for their names (which follow the same name as template object but with different numbered ends). Therefore, it is not possible to distinguish each pallet from the simulation level, under which circumstance a data block dedicated to pallet number and their positions is created.

By giving number of each cell shown in Figure 31, the position of each pallet is illustrated by the cell number. Cell number 5, 4, 11, 13, 20 and 21 are the decision points. While the pallet is being transported on the production line, the data block will be updated with their new cell numbers.

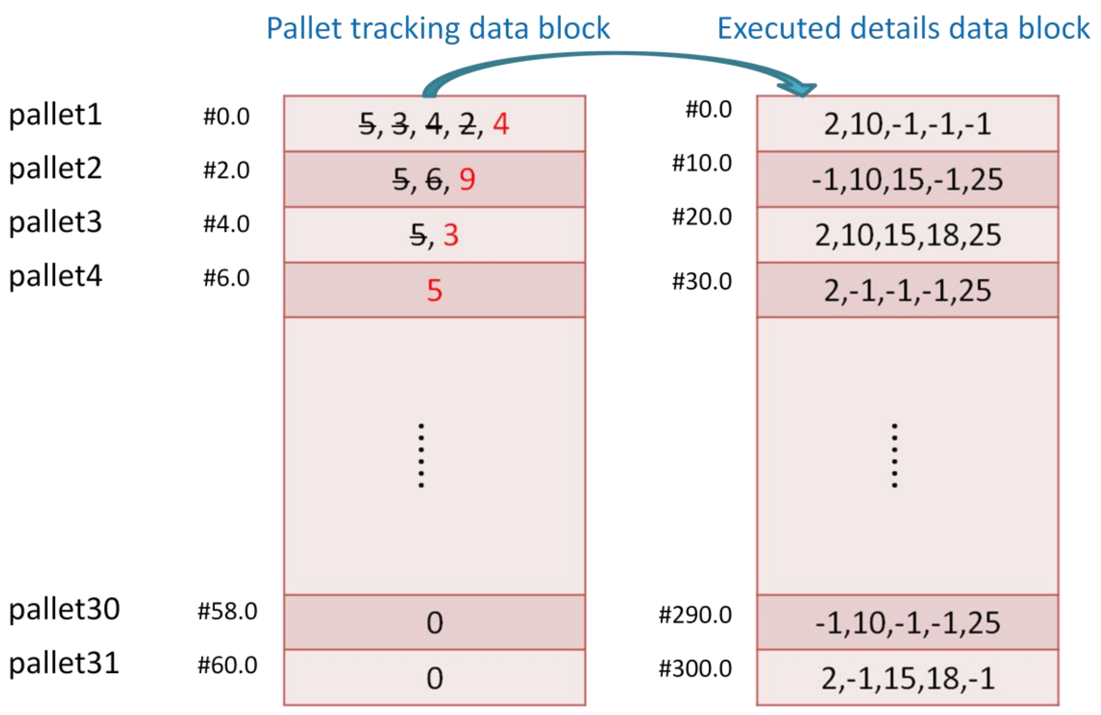


Figure 34 Pallet-position data block

Figure 34 shows the data block principle for recording pallet position and the relation between itself and the executed data block. When the program scans and finds the current cell is decision point, it recognizes the pallet number first and then turns to the executed data block to find the recipe for current pallet and by identifying the decision point cell number, the corresponding recipe detail is detected and will affect the decision finally. For example, when reaching cell number 11, it is for the decision to go to rightline or not, and the third element of recipe detail determines that; and if the third integer of its recipe equals to 15, then the pallet goes to rightline, vice versa. The red number is the position of current moment. A new function (FC 9) was designed for updating cell numbers for this requirement, which requires the previous cell number to locate which pallet's position is to be updated and current cell number to replace the pallet position. The logic diagram of this function is shown in Figure 35.

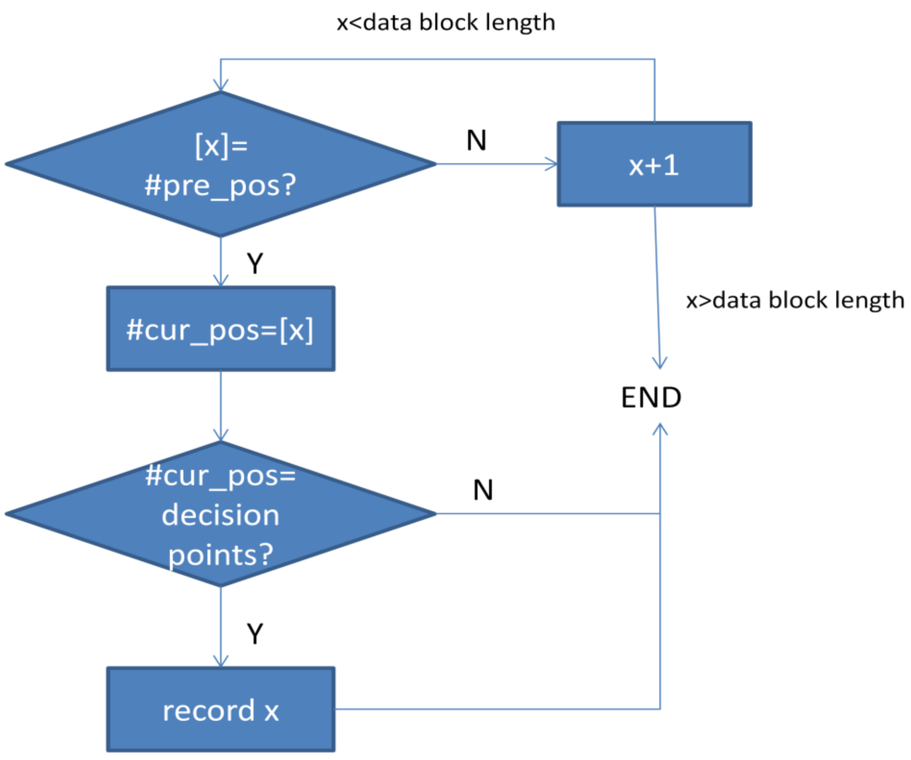


Figure 35 Function logic for updating pallet position

After tracking down the pallet’s recipe, another function (FC 3-8, each for one decision point) which is responding to making direction decision is triggered under the condition of reaching decision cell and knowing the recipe detail for current decision. These functions are connected to the decision signals for each segment demonstrated in section 4.2.4 and initiate the commensurate conveyors. Therefore, if the second element of recipe detail equals to 10 when the position of pallet is 5, for example, it means pallet needs to be loaded to the workstation of start segment. In order to avoid traffic crush, when another pallet is detected at workstation, the current pallet will instead be loaded to the other route and leave the process procedure to the next cycle until all processes needed for this pallet is finished.

4.5. Process perfection

For some cases, all stations needed for a pallet cannot be approached in one cycle, which requires the pallet for another or even more cycles. Thus updating the processed situation of pallets is rather essential for the flexibility of production line.

4.5.1. Process scan

A new function (FC 11) was created for marking finished processes in real-time for the program. When the sensor of the process station is found to be occupied, the program reads into the pallets’ position data block and find out which pallet it is being processed. Then open the data block of executed recipe detail and replace the corresponding recipe detail with -1. Therefore, if the pallet is reaching the same decision point next cycle, the controller is able to realize that it has already taken the process, preventing from repeat work.

4.5.2. Final scan

How to determine whether a pallet finishes all its processes? An idea was employed in the model as to unload the finished pallet out of the line from the last cell of a full cycle, which is s_dl_3, cell number 38.

In the simulation model, a trashcan mechanism was inserted to realize unloading functionality for unloading the pallet. One of the gripper points of trashcan is located next to the sensor of cell 38 which is where the pallet is vanished from.

On program level, when the pallet stops in cell 38, the program reads the position of current pallet first and then scans in the executed recipe detail data block for the processed situation for current pallet. If all five elements are -1, which means the pallet finished all its desired procedure, trigger the trashcan mechanism; if not all five elements are -1, which means the pallet still needs some more process from at least one process station, pallet continue moving to the start lifter. A new function (FC 12) was established to realize this final scan job.

For the pallet position data block, if a pallet finished all the processes and finally reaches to cell 38, the position of current pallet will be replaced with -1, symbolizing a fully processed pallet and distinguishing from other pallets; if the pallet moves back to the start point, preparing for the next cycle, the program scans in the pallet position data block for cell number 38, and replace it with the start cell number 1, which follows the same pallet position of its original one.

The functionality of each customized data block, function in the program and their codes can be found in Appendix 2.

5. ASSESSMENT ANALYSIS

With the simulation model creation ready in chapter 3 and PLC program ready in chapter 4, it is time to run the simulation and retrieve actual appealing measurements from the simulation. One can obtain the recipe execution time and KPI analysis in this chapter. These results are going to be sorted into different levels and analyzed separately according to their properties and degree of importance. System analyses are discussed based on KPI of production performance.

5.1. Simulation results

On process level, cycle time and throughput capacity are the most intuitive numerical indicators that can be measured from a production line. Due to the fact that there are plenty of recipe options for current assembly line, all the possibilities are considered and listed in the tables in following sections. All these route possibilities were simulated under the condition that conveyor speed is set to be 100mm/s. Start point of the pallet is on the lower layer of startlifter, (-830, 7.5, 530), and the counting of cycle time stops when the pallet returns to the start point. Also the process time consumptions are not included in the assessment result.

5.1.1. Single recipe execution

By running the simulation model to each pre-determined recipe, cycle time, throughput per 10mins are collected and listed in the following tables. In the table, '×' means the pallet follows this route, Due to the fact that the simulation was executed by first starting recording time and then setting the start signal which initiates the whole process, there might be some time differences in between; also at some decision point, a little push is needed for the pallet to keep moving, thus some time differences exist in between. Thus the total execution time is calculated with 2s difference.

Table 9 Recipe composition and execution time for single recipe

Recipe number	Robotcell	S_workstation	Rightline	Leftline	E_workstation	Cycle time (±2s)
1	×					127s
2		×				131s
3			×			124s
4				×		150s
5					×	210s
6	×	×				258s
7	×		×			138s
8	×			×		164s
9	×				×	224s
10		×	×			142s
11		×		×		169s
12		×			×	229s
13			×	×		160s
14			×		×	222s
15				×	×	248s
16	×	×	×			269s
17	×	×		×		296s
18	×	×			×	356s
19	×		×	×		174s
20	×		×		×	238s
21	×			×	×	262s
22		×	×	×		178s
23		×	×		×	241s
24		×		×	×	267s
25			×	×	×	258s
26	×	×	×	×		306s
27	×	×	×		×	367s
28	×	×		×	×	393s
29	×		×	×	×	272s
30		×	×	×	×	277s
31	×	×	×	×	×	405s

Some comparison information about cycle time can be summarized from the table above. For example, if the pallet follows the same route for the latter two segments, time duration of taking workstation route is 4-6s longer than robot cell route and 18-20s longer than merely mainline route; if the pallet follows the same route for start and end segment, the longest time was spent when choosing first rightline then leftline in middle segment, which is 9-11s longer than the route first mainline then leftline, 34-36s longer than the rout first rightline then mainline and 47-48s longer the merely mainline route; for the end segment, taking workstation route is generally 23-26s longer than taking mainline route and 97-100s longer than returning to start segment from intermediate lifter.

For the recipes which desire robotcell and s_workstation both cannot be executed during single cycle. Thus the regulations and relations among certain recipes are summarized in the following table.

Table 10 Recipe relation

Recipe number	Equals to
R16	R1 + R10 or R2 + R7
R17	R1 + R11 or R2 + R8
R18	R1 + R12 or R2 + R9
R26	R1 + R22 or R2 + R19 or R7 + R11 or R8 + R10
R27	R1 + R23 or R2 + R20 or R7 + R12 or R9 + R10
R28	R1 + R24 or R2 + R21 or R8 + R12 or R9 + R11
R31	R1 + R30 or R2 + R29 or R7 + R24 or R8 + R23 or R9 + R22 or R10 + R21 or R11 + R20 or R12 + R19

5.1.2. Multiple recipes execution

The designed simulation logic control is able to execute not only single recipe but also multiple recipes. By pressing the desired recipes for the pallets in order, the controller will record the recipe numbers and store them into a data block. Then translate these recipe numbers into executable direction decisions back to the production line. After the pallet passes over the process stations, the corresponding recipe detail will be set to -1 to imply that it finished processing with certain stations.

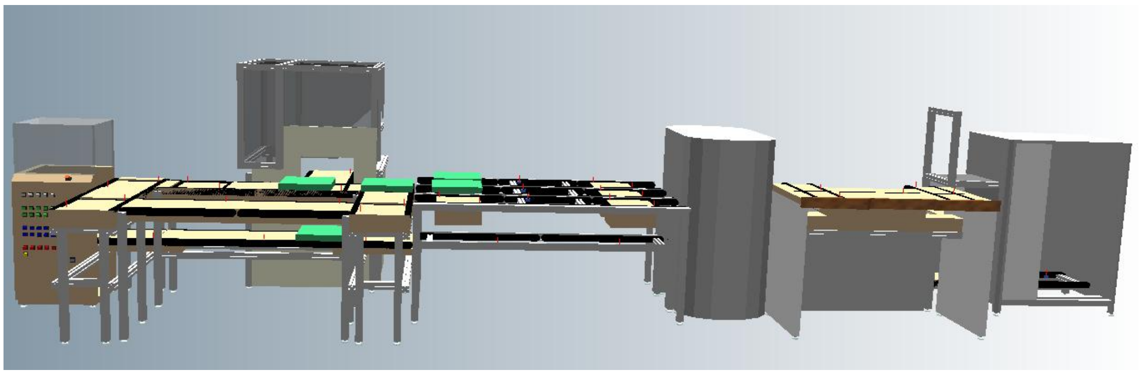


Figure 36 Simulation screenshot

The functionalities of designed simulation program can be summarized as followings:

1. Provide 31 recipe possibilities, including all combinations out of five process stations in the whole line.
2. Specify user interface, simplifying recipe determination with one press.
3. Record customized pallet recipe number, 31 pallets recipe stored in one batch maximum (can be extended).
4. Translate recipe numbers into program readable execution instruction.
5. Locate pallet position in real-time.
6. Make direction decision automatically according to recipe and traffic condition in real-time when facing crossroads.
7. Load same pallet more than one cycle, avoiding traffic crush in process station.
8. Mark finished process, preventing from repeat process.

Taking five pallets in one batch for example, execution time, how many cycles needed to finish processing all pallets and short-term throughput of some combination are listed in the following table.

Table 11 Quantified simulation results of multiple recipes

Recipe order	Execution time	Total cycle number	Throughput within 5mins (unloaded pallets number)
[4, 4, 4, 4, 4]	326s	3 (2 pallets re-cycle)	4
[10, 11, 11, 20, 20]	352s	3 (2 pallets re-cycle)	3
[5, 9, 12, 21, 31]	517s	3 (2 pallets re-cycle)	2
[3, 8, 15, 18, 28]	462s	4 (3 pallets re-cycle)	2
[18, 15, 3, 28, 8]	546s	3 (2 pallets re-cycle)	3
[28, 18, 15, 8, 3]	512s	4 (3 pallets re-cycle)	1

With more experiment result, one can find that even though the recipe numbers are the same, with different order of pallet sending the results can be distinguished.

5.2. KPI analysis

According to Rakar's derivation methodology of KPI [Rakar et al, 2004], production related performance indicators are sorted in a three-level hierarchical structure (Figure 37). Level 1 is characterised by safety and environment in the sense of conformance with regulations and standards; level 2 is composed of indicators related to quality, efficiency and production plan tracking and level 3 deals with issues related to manpower requirements.

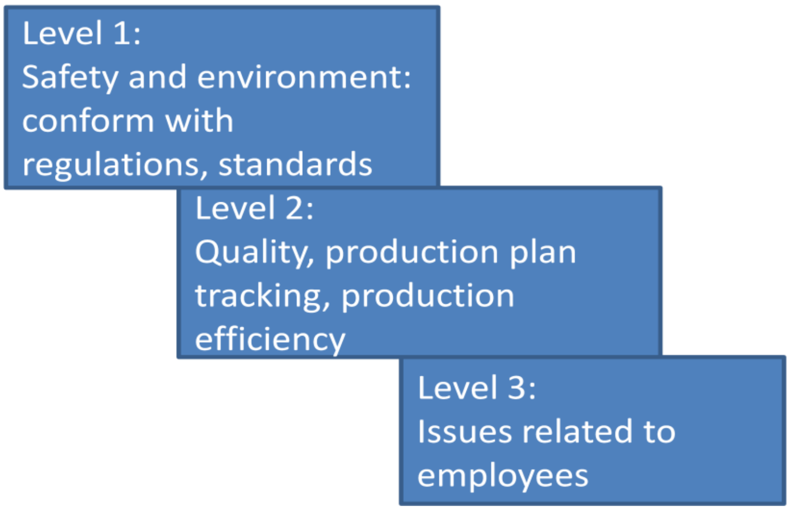

Level 1:
Safety and environment: conform with regulations, standards
Level 2:
Quality, production plan tracking, production efficiency
Level 3:
Issues related to employees

Figure 37 KPI framework [Rakar et al, 2004]

Regarding to the first level, areas referring to work safety and accident, raw material consumptions and their interactions with environment are taken into consideration, such as fresh water consumption, number of hazardous alarms or waste generated before recycling. Unfortunately, in current simulation environment only the transportation process is simulated with no respect to the production process that could be executed in any cell, thus there are no variants related to interactions with environment. Besides the environment factor, it is as well not possible to analyze how safe the process is and how often accident occurs.

For the second level, efficiency variant can be divided into several categories according to plant structure and other aspects, such as efficiency of employees in production, unit production time, energy efficiency and so on. By running current simulation model in FESTO CIRSO Studio, it is easy to record unit production time (cycle time) by terminating the simulation process manually. The drawback of this recording method is the lack of accuracy. It would be nice to have a measure system that can set the starting position and finishing position of a simulation process and be able to calculate cycle time automatically.

From the production plan tracking and quality points of view, the current simulation model appears somehow weak to acknowledge these kinds of information. For the current version of simulation, it is only capable of imitate transportation system of production line off-line. Once the interface has been built between simulation software and actual production line I/Os, it will be possible for the user to monitor the real-time production process from simulation software window and collection of production plan tracking and quality parameters are accessible.

What could be retrieved from the simulation with respect of level 3 indicators is that human intervention is definitely mandatory for assembly line, for the purpose of loading recipes (choosing pallet route) and sending pallets from the starting point. Although it appears that during simulation pallets need a little push when suddenly changing their flowing direction at times, it is actually a software constraint which causes such kind of a problem. However, other factors belonging to level 3, such as turnover rate and complete job satisfaction of employees, require actual embodiment and further investment of current production simulation.

6. CONCLUSION

The focus of this paper is on creating a pallet-based simulation model of a production line customized by Flexlink in FESTO CIROS Studio and analyzing the production performance based on the KPIs retrieved from the simulation. The essential problem is to integrate the simulation model with virtual PLC and realize the transportation logic of the production system.

FESTO CIROS Studio is a proper choice of 3D simulation software for current thesis topic. The software provides profuse libraries with well-made models and mechanisms, from FESTO system platform, robot to conveyors, sensors, from which most mechanisms needed in the assembly line can be found. Yet there are still some limitations, for instant, pallet cannot move independently when changing its conveying direction; end time of a simulation process cannot be set automatically. These constraints lead to a same problem which is the measurements of timing field are not accurate enough.

All indicators that can be obtained from current simulation model are mainly concentrated on process level and a few on production level. This means that FESTO CIROS Studio is designed to concentrate on process simulation, such as transportation simulation, sensor simulation and robotics simulation. Due to the fact that model library provides mechanism of actuator instead of motors, energy consumption and efficiency and other environment relevant factors are not accessible by merely running simulation.

During simulation, it has been a major flaw of program manager of FESTO CIRSO Studio that it does not support real-time monitor on PLC I/Os directly in diagram but only from controller in statement form, which is not easy to read. PLC program needs to be created by SIMATIC manager first and then integrated with simulation controller. However during simulation, it is not possible to observe I/O changing from interface, which brought a lot challenges to the job.

Although the creation of production model is quite satisfying, the generated object that needs to be produced (pallet) cannot be distinguished from each other on model level. The identification of pallets can only be realized within S7 program by mapping its previous and current position. It would be more convenience to provide identification system like radio frequency technology as to allocate pallets with its desired recipe.

Leaving aside former limitations, it is a huge advantage that 3D model file done in FESTO CIROS Studio is rather small, in case of current simulation, around 3MB. This made it possible for users to discuss via email, which was done between the author and product manager of Festo Didactic GmbH in Germany who has been tremendously helpful during this thesis work.

Future work from current thesis topic could be extend the simulation field to include other system of the line, such as robot cell process and power consumption area, to ensure the simulation result more useful for production performance analysis and business level decision making as well. From the perspective of optimization, it is a possible extension to investigate on an algorithm which aims to minimize execution time when loading multiple pallets by sorting pallet loading order. Meanwhile, it is also worth to consider employing other simulation and model methodology to get comparison result of system properties, such as Petri Net modelling or Agent-Based Simulation.

REFERENCES

Banks, J., 1998. Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice, John Wiley and Sons, New York, NY.

Colombo, A.W., 2010. Lecture Notes: Automation Technologies for Intelligent Manufacturing Systems, Tampere, Finland.

Incontrol, 2011. Tutorial Enterprise Dynamics 8, http://support.incontrolsim.com/en/ed-tutorials.html

Korotkin, S.; Zaidner, G.; Cohen, B.; Ellenbogen, A.; Arad, M.; Cohen, Y., 2010. A Petri Net formal design methodology for discrete-event control of industrial automated systems, IEEE 26th Electrical and Electronics Engineers in Israel, pp431-435.

Leitao, P., 2009. Agent-based Distributed Manufacturing Control: A state-of-the-art Survey, Engineering Applications of Artificial Intelligence, v22 n7, pp 979-991.

Lloyd, S.; Yu, H.; Konstas, N., 1995. FMS scheduling using Petri net modelling and a branch & bound search, Assembly and Task Planning, pp141-146.

McLean, C. and Leong, S., 2001. The Expanding Role of Simulation in Future Manufacturing, Proceedings of the 200 Winter Simulation Conference, December 11th to 14th 2001, Phoenix, USA, pp 1478-1486.

Minas, M. And Frey, G., 2002. Visual PLC-programming using signal interpreted Petri nets, Proceedings of the American Control Conference, v6, pp5019-5024.

Murata, T., 1989. Petri Nets: Properties, Analysis and Applications, Proceeding of the IEEE, v77 n4, pp541-580.

Nordgren, W.B., 2001. Taylor Enterprise Dynamics, Proceedings of the 2001 Winter Simulation Conference, v1, pp269-271.

Rakar, A.; Zorzut S.; Jovan V., 2004. Assessment of production performance by means of KPI, Control 2004, University of Bath, UK.

Salminen, M., 2010. Evaluation of simulation tools for modular assembly systems, Master of Science Thesis, Tampere University of Technology.

Shannon, R.E., 1992. Introduction to simulation, Winter Simulation Conference, pp 65-73.

Smith, R.J., 2003. Survey on the Use of Simulation for Manufacturing System Design and Operation, Journal of Manufacturing Systems, v22 n2, pp157-171.

Visual Components, http://www.simx.co.uk/software/vc/Brochures/3DRealizeBrochure.pdf

Yilmaz, O.S and Davis, R.P., 1987. Flexible Manufacturing Systems: Characteristics and Assessment, Engineering Management International, v4 n3, pp 209-212.

Young, C. and Greene, A., 1986. Flexible Manufacturing Systems, American Management Association, New York.

Zhou, M. and Venkatesh, K., 1999. Modelling, simulation, and control of flexible manufacturing systems: a Petri net approach, World Scientific Publishing, Singapore.

Zhu, J.J. and Denton, R.T., 1988. Timed Petri nets and their application to communication protocol specification, Military Communications Conference, v1, pp195-199.

APPENDIX 1

		Single Conveyors	*s_dl_c2 s_dl_c1 m_dl_c2 m_dl_c1 m_ml_c2*	m_rl_c3 m_rl_c1 m_ml_c3 (DP) m_ml_c1 (DP)				LIFTERS			Double Conveyors				End line			
I/O		PATTERN	*e_rl_c4 e_ml_c2*	m_ll_c3 m_ll_c1	m_rl_c2 (S4) m_ll_c2 (S3)	e_dl_c1	s_ws_r1 (s5)	s_ml_l1	m_ml_l1	e_ml_l1	PATTERN	s_rl_cc1 (s1)	s_ml_cc3 s_ml_cc1 (DP)	s_ml_c2 (DP)	e_rl_c3	e_rl_c2 (s5)	e_rl_c1	e_ml_c3 e_ml_c1 (DP)
IX0	0	sensor_b1	sensor_b1	sensor_b1	sensor_b1	sensor_b1	sensor_b1	sensor_b2	inv_ready	PALLET_TRANS	f_sensor_b1	f_sensor_b1	f_sensor_b1	f_sensor_b1	CC_UP	CC_DOWN	SENSOR_B1	SENSOR_B1
	1	sensor_b2		sensor_b2		sensor_b2		sensor_b1	inv_pos_reached	SENSOR_B2	f_sensor_b2		f_sensor_b2	f_sensor_b2	CC_DOWN	CC_UP	TABLE_1	CC_DOWN
	2	crossc_up		crossc_up				pallet_trans	sensor_b1	SENSOR_B3	f_xconv_up		f_xconv_up	f_xconv_up	SENSOR_B1	SENSOR_B1	TABLE_2	MC_UP_RIGHT
	3	crossc_down		crossc_down				pallet_trans_rear	sensor_b2	SENSOR_B1	f_xconv_down		f_xconv_down	f_xconv_down		SENSOR_B2		MC_UP_LEFT
	4	send_1			send_1			lift_down	reset	LIFT_DOWN	send_button	send_button				SEND		
	5	send_2			send_2			lift_up	up_ss	ALARM_UP	mode	mode						
	6	mode			mode				down_ss	ALARM_DOWN								
	7								up_button	LIFT_UP	split_conv	split_conv		split_conv				
IX1	0								down_button	UP_BUTTON	b_sensor_b1	b_sensor_b1	b_sensor_b1	b_sensor_b1				

	1								servi-ce_switch	DOWN_BUTTON	b_sensor_b2		b_sensor_b2	b_sensor_b2				
	2								shaft_A	RE-SET_BUTTON	b_xconv_up		b_xconv_up	b_xconv_up				
	3								shaft_B	RUN_SERVICE	b_xconv_down		b_xconv_down	b_xconv_down				
	4																	
	5																	
	6																	
	7																	

QX 0	**0**	conv_fwd	conv_fwd	conv_fwd	conv_fwd	conv_fwd	conv_fwd	lo-wer_lifter	pos_0	MC_FWD	f_xconv_right		f_xconv_right	f_xconv_right	MC_RUN	MC_RUN	MC_RUN	MC_UP
	1	crossc_left		crossc_left				raise_lifter	pos_1	MC_REV	f_xconv_left		f_xconv_left	f_xconv_left	CC_RUN	CC_RUN	TAB-LE_UP	MC_DOWN
	2	lifter		lifter				conv_fwd	inv_enable	MC_IN_RUN	f_xconv_raise		f_xconv_raise	f_xconv_raise	CC_LIFT	CC_LIFT	TAB-LE_DOWN	MC_RUN
	3	crossc_right		crossc_right				conv_bwd	inv_cal_req	MC_OUT_RUN	f_xconv_lower		f_xconv_lower	f_xconv_lower		SEND_LIGHT		CC_RUN
	4	conv_bwd					conv_bwd		pos_mode	LIFTER_UP	conv_fwd	conv_fwd	conv_fwd	conv_fwd				
	5								run_req	LIF-TER_DOWN	conv_bwd		conv_bwd					
	6								jog_spd	LOW_SPEED	b_conv_fwd	b_conv_fwd		b_conv_fwd				
	7								jog_dir	HIGH_SPEED								
QX 1	**0**								pos_2		b_xconv_right		b_xconv_right	b_xconv_right				
	1								M1_bwd		b_xconv_left		b_xconv_left	b_xconv_left				
	2								M1_fwd		b_xconv_raise		b_xconv_raise	b_xconv_raise				
	3								light_turret		b_xconv_lower		b_xconv_lower	b_xconv_lower				
	4																	
	5																	
	6																	
	7																	

APPENDIX 2

Block name	Major functionality
DB 1-31	Recipe details of recipe 1-31
DB 32	Recipe numbers of each pallet in order
DB 33	Executed recipe details of each pallet in order
DB 34	Pallet positions updated in real-time in order
FC 1	Record the pressed button numbers in DB 32
FC 2	Copy the recipe details from DB 1-31 according to the recipe number and order in DB 32, then paste them in DB 33 in order
FC 3	Read the second element of each recipe detail and make decision of s1 and make decision of direction
FC 4	Read the first element of each recipe detail and make decision of s2 and make decision of direction
FC 5	Read the third element of each recipe detail and make decision of m1 and make decision of direction
FC 6	Read the fourth element of each recipe detail and make decision of m2 and make decision of direction
FC 7	Read the fifth element of each recipe detail and make decision of m3 and make decision of direction
FC 8	Read the fifth element of each recipe detail and make decision of e1 and make decision of direction
FC 9	Load cell number 1 to each new pallet's first position
FC 10	Scan for the previous position of pallet and replace with its current position; if decision points are identified, save the pallet number of current pallet as the reference for FC 3-8
FC 11	Mark the finished element with -1
FC 12	Check if all five elements of current pallet's recipe are -1 and if they are, vanish the pallet and put -1 at position data block; if not, recycle to start point for next cycle

FC 1

```
L     "counter_press"
T     "recipe_order_int"
L     "recipe_order_int"
L     1
-I
T     "recipe_order_int"
L     "recipe_order_int"
ITD
T     "recipe_order"
OPN   "exec"
L     "recipe_order"
L     P#2.0
*D
T     "recipe"
L     #r_num
T     DBW ["recipe"]
L     "recipe"
L     P#60.0
==D
=     "m_comp8"
A     "m_comp8"
BEC
```

FC 2

```
      L     P#0.0
      T     "cur_r"
m001: OPN   "exec"
      L     DBW ["cur_r"]
      T     #recipe_num
      L     #recipe_num
      L     -1
      ==I
      =     "m_comp7"
      A     "m_comp7"
      BEC
      L     #recipe_num
      T     "recipe_num"
      OPN   DB ["recipe_num"]
      L     DBW    0
      T     "r1_temp"
      L     DBW    2
      T     "r2_temp"
      L     DBW    4
      T     "r3_temp"
      L     DBW    6
      T     "r4_temp"
      L     DBW    8
      T     "r5_temp"
      OPN   "exec_detail"
      L     "cur_r"
      L     L#5
      *D
      T     "cur_d"
      L     "r1_temp"
      T     DBW ["cur_d"]
      L     "cur_d"
      L     P#2.0
      +D
      T     "cur_d"
      L     "r2_temp"
      T     DBW ["cur_d"]
      L     "cur_d"
      L     P#2.0
      +D
      T     "cur_d"
      L     "r3_temp"
      T     DBW ["cur_d"]
      L     "cur_d"
      L     P#2.0
      +D
      T     "cur_d"
      L     "r4_temp"
      T     DBW ["cur_d"]
      L     "cur_d"
      L     P#2.0
      +D
      T     "cur_d"
      L     "r5_temp"
      T     DBW ["cur_d"]
      L     "cur_r"
      L     P#2.0
      +D
      T     "cur_r"
      LOOP  m001
```

FC 3

```
OPN   "exec_detail"
L     "pointer_s1"
L     L#5
*D
T     "cur_d2"                //current decision
L     "cur_d2"
L     P#2.0
+D
T     "cur_d2"
L     DBW ["cur_d2"]
L     0
==I
=     "m_comp9"
A     "m_comp9"
R     "s1_ws"
R     "s1_ml"
BEC
AN    "m_comp9"
L     DBW ["cur_d2"]
L     10
==I
=     "m_comp2"
A(
A     "m_comp2"
AN    "s_ws1"
AN    "s_ws2"
AN    "s_2.2_m"
)
=     "s1_decision"
A     "s1_decision"
=     "s1_ws"
AN    "s1_decision"
=     "s1_ml"
```

FC 4

```
OPN   "exec_detail"
L     "pointer_s2"
L     L#5
*D
T     "cur_d1"                //current decision
L     DBW ["cur_d1"]
L     0
==I
=     "m_comp10"
A     "m_comp10"
R     "s2_rc"
R     "s2_ml"
BEC
AN    "m_comp10"
L     DBW ["cur_d1"]
L     2
==I
=     "m_comp1"
A(
A     "m_comp1"
AN    "s_rc"
)
=     "s2_decision"
A     "s2_decision"
=     "s2_rc"
AN    "s2_decision"
=     "s2_ml"
```

FC 5

```
OPN   "exec_detail"
L     "pointer_m1"
L     L#5
*D
T     "cur_d3"                //current decision
L     "cur_d3"
L     P#4.0
+D
T     "cur_d3"
L     DBW ["cur_d3"]
L     0
==I
=     "m_comp11"
A     "m_comp11"
R     "m1_rl"
R     "m1_ml"
BEC
AN    "m_comp10"
L     DBW ["cur_d3"]
L     15
==I
=     "m_comp3"
A(
A     "m_comp3"
AN    "m_r1_m"
AN    "m_r2_m"
AN    "m_r3_m"
)
=     "m1_decision"
A     "m1_decision"
=     "m1_rl"
AN    "m1_decision"
=     "m1_ml"
```

FC 6

```
OPN   "exec_detail"
L     "pointer_m2"
L     L#5
*D
T     "cur_d4"                //current decision
L     "cur_d4"
L     P#6.0
+D
T     "cur_d4"
L     DBW ["cur_d4"]
L     0
==I
=     "m_comp12"
A     "m_comp12"
R     "m2_ll"
R     "m2_ml"
BEC
AN    "m_comp12"
L     DBW ["cur_d4"]
L     18
==I
=     "m_comp4"
A(
A     "m_comp4"
AN    "m_l1_m"
AN    "m_l2_m"
AN    "m_l3_m"
)
=     "m2_decision"
A     "m2_decision"
=     "m2_ll"
AN    "m2_decision"
=     "m2_ml"
```

FC 7

```
OPN   "exec_detail"
L     "pointer_m3"
L     L#5
*D
T     "cur_d5"              //current decision
L     "cur_d5"
L     P#8.0
+D
T     "cur_d5"
L     DBW ["cur_d5"]
L     0
==I
=     "m_comp13"
A     "m_comp13"
R     "m3_c"
R     "m3_r"
BEC
AN    "m_comp13"
L     DBW ["cur_d5"]
L     25
==I
=     "m_comp5"
A(
A     "m_comp5"
AN    "e_m1_m"
)
=     "m3_decision"
A     "m3_decision"
=     "m3_c"
AN    "m3_decision"
=     "m3_r"
```

FC 8

```
OPN   "exec_detail"
L     "pointer_e"
L     L#5
*D
T     "cur_d6"              //current decision
L     "cur_d6"
L     P#8.0
+D
T     "cur_d6"
L     DBW ["cur_d6"]
L     0
==I
=     "m_comp14"
A     "m_comp14"
R     "e_ws"
R     "e_ml"
BEC
AN    "m_comp14"
L     DBW ["cur_d6"]
L     25
==I
=     "m_comp6"
A(
A     "m_comp6"
AN    "e_ws_1"
AN    "e_ws_2"
AN    "e_ws_3"
AN    "e_ws_4"
AN    "e_ws_5"
)
=     "e_decision"
A     "e_decision"
=     "e_ws"
AN    "e_decision"
=     "e_ml"
```

FC 9

```
      L     "counter_1"
      T     "pallet_order_s1_int"
      L     "pallet_order_s1_int"
      L     1
      -I
      T     "pallet_order_s1_int"
      L     "pallet_order_s1_int"
      ITD
      T     "pallet_order_s1"
      L     "pallet_order_s1"
      L     P#2.0
      *D
      T     "pallet_lo"
      OPN   "pallet_location"
      L     "counter_1"
      L     "counter_press"
      <=I
      =     "m_comp43"
      AN    "m_comp43"
      JC    m001
      A     "m_comp43"
      L     1
      T     DBW ["pallet_lo"]
      BEC
m001: L     P#0.0
      T     "pointer_38"
next: L     DBW ["pointer_38"]
      L     38
      ==I
      =     "m_comp44"
      A     "m_comp44"
      JC    m002
      AN    "m_comp44"
      L     "pointer_38"
      L     P#2.0
      +D
      T     "pointer_38"
      L     "pointer_38"
      L     P#60.0
      ==D
      =     "m_comp45"
      A     "m_comp45"
      BEC
      AN    "m_comp45"
      LOOP  next
m002: L     1
      T     DBW ["pointer_38"]
      BEC
```

FC 10

```
    L     P#0.0
    T     "pointer_db34"
    L     #cur_pos
    T     "cur_pos"
next: OPN   "pallet_location"
    L     DBW ["pointer_db34"]
    L     #pre_pos
    ==I
    =     "m_comp15"
    L     "pallet_order_s1"
    A     "m_comp15"
    JC    m001
    A     "m_comp15"
    JCN   m002
m002: L     "pointer_db34"
    L     P#60.0
    ==D
    =     "m_comp22"
    A     "m_comp22"
    BEC
    AN    "m_comp22"
    L     "pointer_db34"
    L     P#2.0
    +D
    T     "pointer_db34"
    LOOP  next
m001: L     "cur_pos"
    T     DBW ["pointer_db34"]
    L     #cur_pos                  // if it is s1
    L     5
    ==I
    =     "m_comp46"
    AN    "m_comp46"
    JC    m009
    A     "m_comp46"
    L     "pointer_db34"
    T     "pointer_s1"              //make copy
    BEC
m009: L     #cur_pos                // if it is s2
    L     4
    ==I
    =     "m_comp24"
    AN    "m_comp24"
    JC    m004
    A     "m_comp24"
    L     "pointer_db34"
    T     "pointer_s2"              //make copy
    BEC
m004: L     #cur_pos                // if it is m1
    L     11
    ==I
    =     "m_comp25"
    AN    "m_comp25"
    JC    m005
    A     "m_comp25"
    L     "pointer_db34"            //make copy
    T     "pointer_m1"
    BEC
m005: L     #cur_pos                // if it is m2
    L     13
    ==I
    =     "m_comp26"
    AN    "m_comp26"
    JC    m006
    A     "m_comp26"
    L     "pointer_db34"            //make copy
    T     "pointer_m2"
    BEC
m006: L     #cur_pos                // if it is m3
    L     20
    ==I
    =     "m_comp27"
    AN    "m_comp27"
    JC    m007
    A     "m_comp27"
    L     "pointer_db34"            //make copy
    T     "pointer_m3"
    BEC
m007: L     #cur_pos                // if it is e1
    L     21
    ==I
    =     "m_comp28"
    AN    "m_comp28"
    JC    m008
    A     "m_comp28"
    L     "pointer_db34"            //make copy
    T     "pointer_e"
    BEC
m008: L     #cur_pos
    L     38
    ==I
    =     "m_comp40"
    AN    "m_comp40"
    BEC
    A     "m_comp40"
    L     "pointer_db34"
    T     "pointer_finish"
    BEC
```

FC 11

```
    L     #cur_pos
    L     2                         //robotcell?
    ==I
    =     "m_comp30"
    A     "m_comp30"
    JC    m001
    L     #cur_pos                  //s_worstation?
    L     10
    ==I
    =     "m_comp31"
    A     "m_comp31"
    JC    m002
    L     #cur_pos                  //rightline?
    L     15
    ==I
    =     "m_comp32"
    A     "m_comp32"
    JC    m003
    L     #cur_pos                  //leftline?
    L     18
    ==I
    =     "m_comp33"
    A     "m_comp33"
    JC    m004
    L     #cur_pos                  //e_workstation?
    L     25
    ==I
    =     "m_comp34"
    A     "m_comp34"
    JC    m005
m001: OPN   "pallet_location"
    L     P#0.0                     //find process station cell
number
    T     "pointer_ps"
n1:   L     DBW ["pointer_ps"]
    L     2
    ==I
    =     "m_comp35"
    A     "m_comp35"
    JC    l1
    AN    "m_comp35"
    JC    r1
l1:   L     "pointer_ps"
    T     "pointer_ex"
```

```
     L     "pointer_ex"
     L     L#5
     *D
     T     "pointer_ex"
     OPN   "exec_detail"
     L     -1
     T     DBW ["pointer_ex"]
     BEC
r1:  L     "pointer_ps"
     L     P#2.0
     +D
     T     "pointer_ps"
     LOOP  n1
m002: OPN   "pallet_location"
     L     P#0.0                    //find process station cell
number
     T     "pointer_ps"
n2:  L     DBW ["pointer_ps"]
     L     10
     ==I
     =     "m_comp36"
     A     "m_comp36"
     JC    l2
     AN    "m_comp36"
     JC    r2
l2:  L     "pointer_ps"
     T     "pointer_ex"
     L     "pointer_ex"
     L     L#5
     *D
     T     "pointer_ex"
     L     "pointer_ex"
     L     P#2.0
     +D
     T     "pointer_ex"
     OPN   "exec_detail"
     L     -1
     T     DBW ["pointer_ex"]
     BEC
r2:  L     "pointer_ps"
     L     P#2.0
     +D
     T     "pointer_ps"
     LOOP  n2
m003: OPN   "pallet_location"
     L     P#0.0                    //find process station cell
number
     T     "pointer_ps"
n3:  L     DBW ["pointer_ps"]
     L     15
     ==I
     =     "m_comp37"
     A     "m_comp37"
     JC    l3
     AN    "m_comp37"
     JC    r3
l3:  L     "pointer_ps"
     T     "pointer_ex"
     L     "pointer_ex"
     L     L#5
     *D
     T     "pointer_ex"
     L     "pointer_ex"
     L     P#4.0
     +D
     T     "pointer_ex"
     OPN   "exec_detail"
     L     -1
     T     DBW ["pointer_ex"]
     BEC
r3:  L     "pointer_ps"
     L     P#2.0
     +D
     T     "pointer_ps"
     LOOP  n3
m004: OPN   "pallet_location"
     L     P#0.0                    //find process station cell
number
     T     "pointer_ps"
n4:  L     DBW ["pointer_ps"]
     L     18
     ==I
     =     "m_comp38"
     A     "m_comp38"
     JC    l4
     AN    "m_comp38"
     JC    r4
l4:  L     "pointer_ps"
     T     "pointer_ex"
     L     "pointer_ex"
     L     L#5
     *D
     T     "pointer_ex"
     L     "pointer_ex"
     L     P#6.0
     +D
     T     "pointer_ex"
     OPN   "exec_detail"
     L     -1
     T     DBW ["pointer_ex"]
     BEC
r4:  L     "pointer_ps"
     L     P#2.0
     +D
     T     "pointer_ps"
     LOOP  n4
m005: OPN   "pallet_location"
     L     P#0.0                    //find process station cell
number
     T     "pointer_ps"
n5:  L     DBW ["pointer_ps"]
     L     25
     ==I
     =     "m_comp39"
     A     "m_comp39"
     JC    l5
     AN    "m_comp39"
     JC    r5
l5:  L     "pointer_ps"
     T     "pointer_ex"
     L     "pointer_ex"
     L     L#5
     *D
     T     "pointer_ex"
     L     "pointer_ex"
     L     P#8.0
     +D
     T     "pointer_ex"
     OPN   "exec_detail"
     L     -1
     T     DBW ["pointer_ex"]
     BEC
r5:  L     "pointer_ps"
     L     P#2.0
     +D
     T     "pointer_ps"
     LOOP  n5
```

FC 12

```
     OPN   "exec_detail"
     L     0
     T     "scan_times"
     L     "pointer_finish"
     L     L#5
     *D
     T     "pointer_copy"
next: L     DBW ["pointer_copy"]
     L     -1
     ==I
     =     "m_comp41"
     AN    "m_comp41"
     BEC
     A     "m_comp41"
     L     "scan_times"
     INC   1
     T     "scan_times"
     L     "pointer_copy"
     L     P#2.0
     +D
     T     "pointer_copy"
     L     "scan_times"
     L     5
     ==I
     =     "m_comp42"
     A     "m_comp42"
     JC    m001
     AN    "m_comp42"
     LOOP  next
m001: =     "disappear"
     OPN   "pallet_location"
     L     -1
     T     DBW ["pointer_finish"]
     BEC
```