Modular Control of a Reconfigurable Conveyer System

by
Marcus Jacobus Kotzé

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Mechatronic) in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof AH Basson

December 2016

DECLARATION

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2016

ABSTRACT

Modular Control of a Reconfigurable Conveyer System

MJ Kotzé
Department of Mechanical and Mechatronic Engineering
Stellenbosch University
Private Bag X1, 7602 Matieland, South Africa
Thesis: M. Eng (Mechatronic)
December 2016

This thesis documents research into the control of a reconfigurable conveyer system. The conveyer forms part of a reconfigurable assembly system. The main reason for research into reconfigurable systems is the need in the modern manufacturing industry for systems to be adaptable to changing product designs. Modern manufacturing requires many short production runs of different products, instead of the traditional long term production run of a standard product.

The objective of this thesis was to design and implement a controller for a reconfigurable conveyer system. The conveyer hardware was fixed, but the control system had to be developed. A distributed control architecture was chosen. The conveyer was sectioned into different modules, and each module was controlled by a separate controller. This modularity enhanced the conveyer's reconfigurability properties.

The control was split into two sections, low level control (LLC) and high level control (HLC). LLC was responsible for the actuation of the conveyer's hardware, using programmable logic controllers (PLCs). The HLC did more advanced tasks such as communication, task management and path planning.

The software design was based on a modified ADACOR holonic architecture. The holonic architecture was used because it promotes reconfigurability. Path planning was done using Dijkstra's algorithm.

Experimental results show that the conveyer was able to successfully handle multiple pallets. A reconfiguration test was done, which showed improved results from previous work. The design mostly exhibits the characteristics of reconfigurability, which are modularity, customizability, scalability, integratibilty and diagnosability.

UITTREKSEL

Modulêre Beheer van ‘n Herkonfigureerbare Vervoerbandstelsel

MJ Kotzé
Departement van Meganies and Megatroniese Ingenieurswese
Universiteit Stellenbosch
Privaatsak X1, 7602 Matieland, Suid-Afrika
Tesis: M. Ing (Megatronies)
Desember 2016

Hierdie tesis dokumenteer navorsing oor die beheer van ‘n herkonfigureerbare vervoerbandstelsel. Die vervoerband vorm deel van ‘n herkonfigureerbare vervaardiging stelsel. Die hoofrede vir die navorsing in herkonfigureerbare stelsels is die behoefte in die moderne vervaardigingsbedryf aan stelsels wat aanpasbaar kan wees vir veranderinge in produk-ontwerpe. Moderne vervaardiging vereis kort produksie lopies van verskillende produkte, in plaas van ‘n tradisionele langtermyn produksie lopie van ‘n standaard produk.

Die doel van hierdie tesis was om ‘n beheerder vir ‘n herkonfigureerbare vervoerbandstelsel te ontwerp en implementeer. Die vervoerband hardeware was vooraf bepaal, maar die beheerstel moes ontwikkel word. ‘n Verspreide beheer-argitektuur is gekies. Die vervoerband is verdeel in verskillende modules, en elke module is deur ‘n afsonderlike beheerder beheer. Die modulariteit van die ontwerp het die herkonfigureerbaarheid van die stelsel versterk.

Die beheer was verdeel in twee gedeeltes, lae-vlak-beheer (LVB) en hoë-vlak-beheer (HVB). LVB was verantwoordelik vir aktueer van die vervoerband se hardeware, deur middel van programmeerbare-logika-beheerders (PLBs). Die HVB het meer gevorderde take hanteer, soos kommunikasie, taakbestuur en roete-beplanning.

Die sagteware-ontwerp was gebaseer op ‘n aangepaste ADACOR holoniese argitektuur. Die holoniese argitektuur was gebruik omdat dit herkonfigureerbaarheid bevorder. Roete-beplanning was gedoen deur die gebruik van Dijkstra se algoritme.

Eksperimentele resultate dui daarop dat die vervoerband veelvuldige pallette suksesvol kon hanteer. ‘n Herkonfigurasie toets is gedoen, wat verbeterde resultate getoon het in vergelyking met vorige navorsing. Die ontwerp vertoon in die meeste opsigte die eienskappe van herkonfigureerbaarheid, naamlik modulariteit, aanpasbaarheid (*flexibility)*, skaleerbaarheid, integreerbaarheid en diagnoseerbaarheid.

To my mother and father.

Thank you for all you have done
for Lara, Christiaan and I.

We are blessed beyond measure
by your love and care.

ACKNOWLEDGEMENT

Thank you to everyone who assisted me during the completion of this thesis. I would like to especially thank the following people:

- Professor Anton Basson. Thank you first of all for giving me this opportunity to take my studies further. Thank you for supervision, wisdom, patience and willingness to share your immense knowledge, experience and skill with me.
- My fellow members of the Mechatronics, Automation and Design Research Group. In particular Karel Kruger, who, with your friendly advice, wisdom and encouragement, was more a second supervisor than a colleague.
- Reynaldo Rodriguez, for your unending patience and technical skill. Your advice and assistance in the convertibility experiment and hardware installation was invaluable.
- To my friends, in particularly my fellow residents of “Studio 5” and the “Dagbreek Aftree Oord.” Thank you for your support, good humour, and for helping make my last two years at university the best they could be.
- My family, for your never ending love, care and support.

Finally, to our heavenly Father, for all the seen and unseen miracles of life that You us give every day.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ADACOR	Adaptive Holonic Control Architecture
AGV	Autonomous Ground Vehicle
ConC	Conveyer Controller Holon
ConMap	Conveyer Map Holon
CNC	Computer Numerical Control
DMS	Direct Manufacturing System
DOF	Degree Of Freedom
FMS	Flexible Manufacturing System
HLC	High Level Controller
HMES	Holonic Manufacturing Execution Systems
HMI	Human Machine Interface
I/O	Input/Output
LLC	Low Level Controller
NC	Numerical Control
OH	Operational Holon
PH	Pallet Holon
PLC	Programmable Logic Controller
PROSA	Product Resource Order Staff Architecture
RFID	Radio-Frequency Identification
RMS	Reconfigurable Manufacturing System
RQA	Reconfigurable Quality Assurance
SH	Supervisor Holon

1 INTRODUCTION

1.1 Background

Automated assembly lines started 100 years ago with Henry Ford's mass produced Model T Ford. This was the start of mass production which made previously unaffordable products available to many more potential consumers. Production lines in the mid 1900s had a very high production rate for a single product, and therefore were highly profitable when the demand was high for that product.

The introduction of NC and CNC machines in the 1970s, lead to the creation of flexible manufacturing systems (FMSs) in the early 1980s. However, due to high initial cost, it took 20 years before FMSs began to make inroads into automobile manufacturing, which is the largest market for FMSs (Koren & Shpitalni 2010).

During the 1990s, globalization led to increased competition between manufacturers, as the playing field had expanded to include multinational companies. At this time, the main objectives of manufacturing companies were productivity, quality and flexibility. FMSs were too expensive and not flexible enough to offer the customization that manufacturers wanted in their products, nor was it efficient for short production runs and fast changeovers to slightly modified or different products, which is required by the modern market.

An example of the loss of production time associated with factory floor changeover is that of the Volkswagen automobile plant in Uitenhage. It took roughly a year for the assembly plant to be retooled from manufacturing Citi Golfs to manufacturing Polos. For some products this is an unacceptably long time to wait before production can be started on a new product.

Reconfigurable Manufacturing Systems (RMSs) were developed to address these issues. RMSs also are potentially suited to industries which have small production volumes and high product variety. This makes RMSs potentially applicable to many South African industries. Most South African industries have been reliant on manual labour and conventional automation practices for their manufacturing. However, with competition increasing from overseas manufacturers which have significantly lower labour rates, RMSs may offer a cheaper and more efficient alternative. The implementation of an RMS in a South African company forms the basis for this thesis.

CBI Electrical: Low Voltage (CBI) is one of the subsidiaries of the CBI Electric Group. CBI Electric Group is owned by Reunert Ltd, a South African company listed on the Johannesburg Stock Exchange. One of CBI's product ranges is circuit breakers, which are small electromechanical devices designed to break an electric circuit if the current is too high. This is to protect other components in the

circuit from excessively high current. CBI currently has an assembly factory in Lesotho to utilize the low labour rates in that country. However, since the assembly of the circuit breakers is done manually, there are related quality control issues. Increased competition from foreign manufacturers has led to CBI looking at ways of automating some sections of the assembly process in order to improve the overall assembly of circuit breakers. This will enable CBI to better compete with overseas manufacturers who can produce circuit breakers at very low prices.

The Stellenbosch University's Mechatronics, Automation and Design Research Group's (MADRG) current focus includes the development of a reconfigurable manufacturing system for circuit breakers. Part of the research is to determine whether automated processes will be more efficient in terms of quality, as well as cost, than the current manual process. The system has to be reconfigurable, meaning that a very low downtime is required when the system is modified to manufacture a different type of circuit breaker.

The current activities of the Research Group focus on the development of a reconfigurable quality assurance (RQA) cell. The cell includes a pallet magazine, a conveyor, and various stations situated around the conveyor belt required for the quality assurance process.

1.2 Objectives

The main focus of the thesis is the control of a reconfigurable conveyor system for the RQA cell. Since the conveyer controller developed by Le Roux (2013) was not able to handle multiple pallets and took approximately two weeks of downtime to reconfigure the conveyer, the objectives of this thesis are:

- To evaluate the feasibility of a new modular control architecture, where each functional module of the conveyer has its own low level controller.
- To evaluate the impact of such a modular controller on the reconfiguration time.

1.3 Scope

The conveyer system must be able to handle multiple pallets. The control program will therefore include traffic management for the conveyor. The thesis will focus on achieving safe and reliable movement of pallets, but the optimisation of the traffic flow will not be included. The optimisation of the system is being considered by two other students at Stellenbosch University. One student will be using simulations to simulate a large conveyer system and test optimisation, while another will be using an ants-based holonic control architecture for the distributed control of the conveyer system.

The research will use on the Bosch-Rexroth TS2plus conveyer system currently available in the Automation Laboratory of Stellenbosch University. The conveyer includes a parallel section, transverse conveyers, lifting units, stop gates, proximity sensors, a pallet magazine and RFID read/write stations.

The conveyer control system must be reconfigurable. An alternative set-up must be achieved with minimal downtime. The control must be able to quickly adapt to the new reconfiguration. The system must also conform to the six characteristics of reconfigurable manufacturing systems, as will be discussed in the literature review.

2 LITERATURE REVIEW

The section will discuss current manufacturing systems, with an eventual focus on reconfigurable manufacturing systems. The holonic control architecture will be considered, and current examples thereof will be presented. Finally, transport systems and reconfigurable transport systems will be discussed.

2.1 Manufacturing Systems

Manufacturing has moved away from low volume, high variety and high human involvement, to high volume production lines which emphasize lean manufacturing. Lean manufacturing means reducing wastage as much as possible, and keeping the manufacturing process as efficient as possible. These are known as Direct Manufacturing Systems (DMS). However, in the 1980s, customization and consumer preference became important. DMSs did not have the flexibility required to achieve this. As a result, manufactures had to start looking for systems which could handle different types of products, yet still have high production volumes at affordable costs (Mehrabi, Ulsoy, & Koren 2000).

The Flexible Manufacturing System (FMS) paradigm was introduced to service this requirement. According to ElMaraghy (2006), there are at least ten types of flexibilities associated with manufacturing systems.

- *Machine Flexibility:* Different types of operations are performed without a change in set-up.
- *Material Handling Flexibility:* Total number of possible routes between machines.
- *Operation Flexibility:* Number of possible processing plans for part production.
- *Process Flexibility:* Set of part types that can be manufactured without a major setup change.
- *Product Flexibility:* Ability (in terms of time and cost) to introduce new products into an existing product mix.
- *Routing Flexibility:* Number of possible routes for all part types.
- *Volume Flexibility:* The ability to change production volumes profitably within installed production capacity.
- *Expansion Flexibility:* Ability (in terms of effort and cost) of changing capacity and/or capability through changing the physical make up of the system.
- *Control Program Flexibility:* The ability of a system to run without interruption due to the high intelligence of the machines and system control software.

- *Production Flexibility:* Number of parts that can be manufactured without adding significant capital equipment.

These characteristics show the different types of flexibility that flexible machines should exhibit.

However, according to Mehrabi *et al*. (2000), FMSs were not entirely successful, due to:

- *Cost:* In many cases the FMS included more functions than what was required. Additional capital equipment was procured with the thought in mind that it might be useful in the future. As a result, expensive and unused equipment taking up floor space became a problem.
- *Utilising inadequate system software:* Developing user specified software for an FMS is highly is expensive; therefore inadequate software was used to offset initial costs.
- *Unreliability:* FMS systems are not highly reliable.
- *Obsolescence:* Because of technology advances and FMS's fixed software and hardware, there was a high risk of an FMS rapidly becoming obsolete.

Reconfigurable Manufacturing Systems (RMS) was introduced to address these shortfalls. An RMS is designed from the outset for quick adjustment of production capacity and functionality by changing or rearranging its components. The key characteristics of an RMS according to Koren *et al*. (1991) are:

- *Modularity:* All major components of an RMS are modular, such as structural elements, axes, controls, software and tooling.
- *Integratibilty:* The machine and control modules are designed with interfaces that make it easy for the components to integrate with each other to form the whole system.
- *Customization:* Customized flexibility which means that machines are built around parts of the production family and provide the flexibility required to build only those parts of the family. Customized control is achieved by integrating control modules with the aid of open-architecture technology.
- *Convertibility:* The system must be capable of fast changeovers between existing products and must be easily adaptable for the production of future products.
- *Diagnosability:* The system must detect faults or unacceptable part quality within itself. This will significantly reduce ramp up time between configuration changes.

ElMaraghy (2006) introduced a sixth characteristic, that of *scalability*, which the ability change the capacity of the system quickly and economically.

Customizability and scalability are the distinguishing characteristics of RMSs. Modularity, integrability and diagnosability is necessary to achieve quick and easy reconfiguration.Robustness (the ability to continue operating effectively in the presence disturbances) is not exclusively related to RMSs, but RMSs' characteristics also promote robustness.

According to Mehrabi*et al*. (2000), RMS should not be more expensive than FMS or DMS. This is because an RMS is installed to the exact production capacity and functionality required, however, it is possible to easily upgrade the system in the future. DMSs, as previously mentioned, have high production capacity, but low product variability. FMSs on the other, exhibit too much product variability functionality. It is often the case that components of an FMS are left standing idle. Figure 1 (adaption from ElMaraghy 2006) illustrates the basic difference between the three production paradigms discussed. While there is some overlap, it would seem that RMS is the middle ground between FMS and DMS.

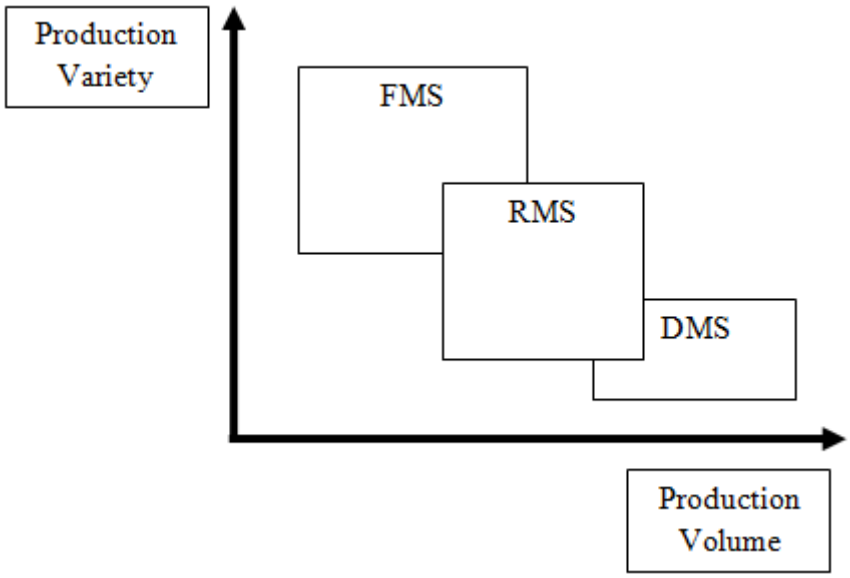


Figure 1: Manufacturing paradigm differences

Within the context of an RMS cell, the conveyor subsystem itself can be considered to be a "major module", but the TS2plus conveyor subsystem is, in itself, also an RMS, with modules that can be rearranged to adapt to changes in production demands. Some subsystems in the RMS cell may be reconfigured to complete different batches during one day, with short conversion times between batches of different product families. However, due to the mechanical, informational and control complexity of the conveyor, its reconfiguration time can be expected to only allow less frequent reconfiguration. The TS2plus system has

excellent modularity and good integrability, but the conveyer control system will be a major determining factor in the transport system's modularity, integrability and diagnosability.

2.2 Control Architectures

There are three main types of control architectures as described by Meng *et al*. (2006). They are centralized, hierarchical and heterarchical. Figure 2 shows three different types of control structures. Note that in Figures 2 and 3, a square symbolises a controlling unit, and a circle symbolises a resource unit.

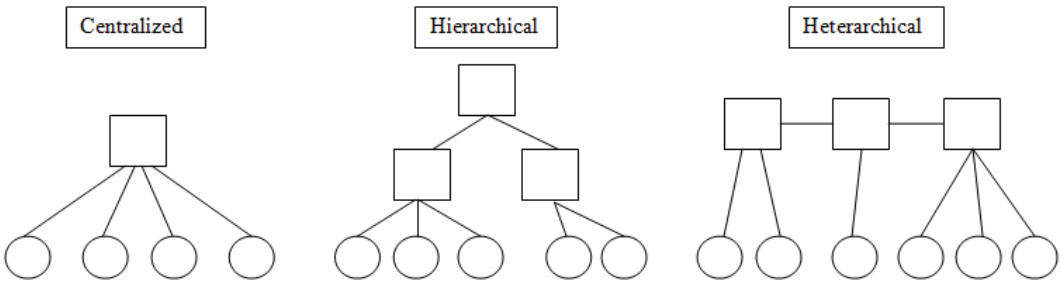


Figure 2: Different control architectures (adapted from Meng *et al.* 2006)

Centralized control consists of one central controller controlling many machine components. It is responsible for all decision making and executing all process. This is type of control is usually implemented in conventional control systems.

Hierarchical control has different levels of control, which has multiple controllers within the system. Instructions are sent downwards, and feedback is sent upwards. This architecture is also used in conventional systems.

Heterarchical systems have no hierarchical levels of control and the controllers operate independently. They are therefore not made aware of the global objective.

Meng *et al.* (2006) states that centralised and hierarchical have disadvantages such as structural rigidity, difficulty of control system design, and lack of flexibility. Meng *et al.* (2006) recommends a hybrid system of hierarchical and heterarchical as shown in Figure 3.

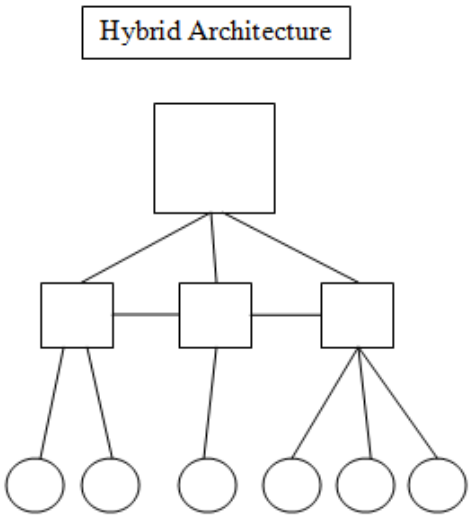


Figure 3: Hybrid control architecture

This system keeps some hierarchical control in order to maintain global performance. However, the distributed controllers can also operate independently and communicate with each other. This type of hybrid control, or strictly heterarchical control is typically used in holonic control systems, which is discussed in Section 2.3.

2.3 Holonic Control

Traditional (centralised) control of a conveyer does not provide the modularity, integrability and diagnosability required in RMS. Alternative control architectures are therefore needed, such as the hybrid control system. Holonic control architectures, although originally developed to improve robustness, are often used in RMSs.

A holon is a component of a complex system that, while contributing to the function of the system as a whole, demonstrates autonomous, stable and self-contained behaviour or functions.

A holon has the following characteristics (Versteegt & Verbraeck 2002):

- *Autonomy*: Holons can create their own plans and control how these plans are carried out.
- *Cooperation:* Holons can communicate and cooperate with other holons in order to achieve their goals. They can also exchange and make mutually acceptable plans. Holons often require the service of other holons in order to achieve their goals.

- *Goal-orientated:* A goal for a holon is a state which must be achieved by that holon. Each holon must have a goal, and the various goals of the holons in a system make up the various sub-goals of the entire system.
- *Control and representation of physical aspects*: A holon can consist of control components and it can act as a representation of a physical resource.

In manufacturing or assembly systems, a holon is an autonomous and cooperative building block for transforming, transporting, storing or validating information of physical objects (Paolucci & Sacile 2005). There are two widely referenced holonic control architectures in manufacturing, PROSA and ADACOR.

PROSA (Product-Resource-Order-Staff-Architecture) has four classes of holons (Van Brussel *et al*. 1998):

- Product
- Resource
- Order
- Staff

The resource holon corresponds to a resource in the manufacturing system. A resource can be a related to a machine, such as a CNC machine or conveyer belt system. The product holon corresponds to a product type, and contains instructions as to how instances of the product should be manufactured using the resources available. An order holon corresponds to a task that needs to be executed. It manages the resource allocations required to produce an instance of a product. It does this by consulting with the product holon to determine what operations are required and then searches for suitable resources to perform those operations. The staff holons supports the manufacturing operation. PROSA is more associated with heterarchical control, because of the distributed control with little or no hierarchical control.

The other architecture for holonic control is ADACOR (ADAptive holonic Control aRchitecture) which employs four classes of holons (Leitão & Restivo 2006):

- Product
- Task
- Operational
- Supervisor

The product, task and operational holons in ADACOR are very similar, respectively, to the product, order and resource holons in PROSA. However,

ADACOR's supervisor holon has no counterpart in PROSA, and is responsible for introducing coordination and global optimisation in a system with decentralized control. This is similar to the hybrid control scheme introduced in Section 2.2.

In holonic control, one of the key notions is to have a close relationship between the control architecture and the system hardware. Typically, therefore, each "major component" (in terms of RMS terminology) will be associated with its own resource or operational holon. In this way, holonic control improves the modularity of the control system. Further, the ability of holons to negotiate with each other to achieve individual and common objectives allows holonic control systems to autonomously find feasible (if not optimum) procedures to complete production steps, even after reconfiguration or disruption in parts of the manufacturing system. Holonic systems therefore have attractive integrability properties.

In the case study considered for this thesis, autonomous reconfiguration is not seen as a priority. However, with a holonic control system, the steps required to reconfigure a control system after many types of reconfiguration, will be within the means of personnel with moderate skill levels. The reconfiguration (including ramp-up testing) should also require significantly less time to complete.

Another feature of holonic control is that it does allow a sense of hierarchy, which Leitão and Restivo (2006) call its "fractal nature" (illustrated in Figure 4). In these terms, the manufacturing cell as a whole can be seen as an operational holon, when viewed from the perspective of the manufacturing system as a whole. The controller of the cell itself can also adopt a holonic architecture, with its own operational, supervisor and other holons. The cell's more complex major subsystems (such as the conveyor subsystem), while they are parts of operational holons in the cell holarchy, can similarly have their own internal holarchies.

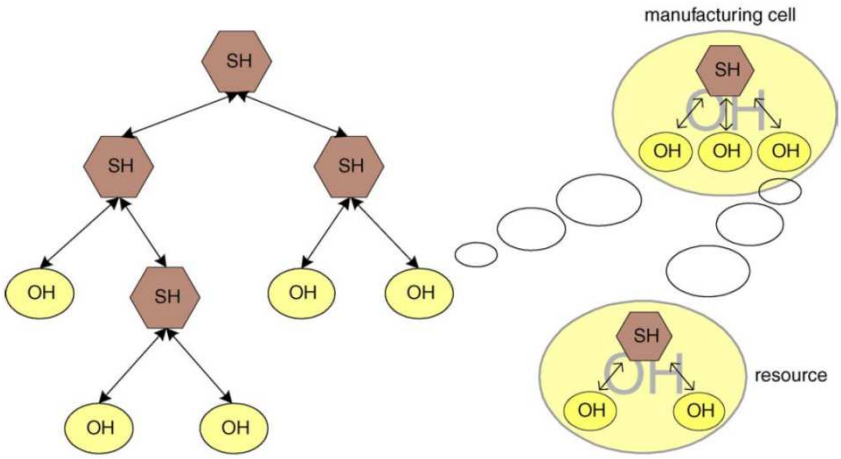


Figure 4: Fractal nature of holonic control (Leitão & Restivo 2006)

2.4 Manufacturing Cell Transport Methods

The three popular types of transport systems are conveyer systems, mobile robots (also called Autonomous Guided Vehicles, or AGVs) and immobile robots.

Immobile robots are appropriate when the required working volume is relatively small. The RMS cell show in Figure 26: Holonic laboratory setup, uses such robots, i.e. the 6 DOF articulated arm and a three axis Cartesian robot, to serve as transport subsystems within their stations.

An AGV is a robot which is programmed to move between stations in a manufacturing cell with a payload. AGVs offer an advantage over conveyer systems in that they are more adaptable, take up less floor space and if one breaks down, it can easily be replaced and production can continue, provided there are more AGVs available. Disadvantages of AGVs include their need to be recharged (RMS applications require easily changeable paths and therefore the use of energy transfer systems along the AGV path is usually infeasible) and a complicated control system if many AGV-to-AGV interactions have to be provided for (such as for intra-cell transport). The costs of an AGV and a conveyer system depend on the system and quality required, and varies depending on the manufacturer and type. It is therefore difficult to compare the systems based on cost.

Conveyer systems are better suited to a confined, high production rate system, such as that of the RQA cell. Modern conveyers are also modular, efficient and more sophisticated than early conveyers, allowing them to be highly reconfigurable. AGVs would be more beneficial in an environment where extremely high flexibility is required, and where the floor space is larger. They are also more useful in an environment where a fixed structure, such as a conveyer, would hamper operations (Weber 2008).

It would be interesting to compare the use of AGVs with conveyer systems, but since the current laboratory setup makes use of a conveyer system, it was chosen as the focus for the thesis.

2.5 Current and Potential Conveyer Control Strategies

The following section discusses some of the conveyer control strategies currently used and experimented with by other researchers.

2.5.1 Previous Work at Stellenbosch University

Le Roux (2013) evaluated different control implementation approaches for the conveyor control, with a focus on enhancing reconfigurability. The study was also done on the Bosch-Rexroth TS2plus conveyer system. A Siemens S7 PLC was used as a centralised low level controller.

The study compared IEC61311-3, IEC64199 and agent-based control implementations with each other. Data tables were used to encapsulate the transitions required to move a pallet from one station on the conveyor to another. The data tables not only captured the particular configuration, but also facilitated communication between high-level and low-level controllers.

The study concluded that IEC61311-3 and IEC64199 architectures are advantageous in lower levels of control. However, the low maturity of development environments for the IEC64199 function blocks was a limitation. Agent based systems offer reliable control and powerful communication tools, but requires a higher level of expertise than the IEC64199 function blocks. For the control of high level reconfigurable manufacturing systems, it was concluded that agent-based control is superior to IEC64199 function blocks (Le Roux 2013). One of the objectives was also multiple pallet control. However, this was not completely achieved. To reconfigure the system also took approximately two weeks of downtime, which is something which can be improved upon.

Le Roux (2013) recommended further work on robust multi-pallet transportation. He also recommended a more distributed or heterarchical control system be used instead of a centralised system.

2.5.2 Holonic Manufacturing Execution Systems (HMES)

Van Belle *et al*. (2009) presented a conveyer control system based on a holonic manufacturing execution system (HMES). Their system was implemented using agents and was based on the PROSA reference architecture.

In their system, the interaction between holons is based on natural systems and, more specifically, that of an ant colony. In an ant colony, ants deposit information in the form of pheromones which other ants can use. In this control strategy, the PROSA holons create lightweight holons, or ant holons, for similar purposes. The two types of ants are exploring ants and intention ants. These ants are sent out by the order holons. The exploring ants will then report back possible solutions or changes in the system. Intention ants can then be created to reserve capacity on the required resources.

With this strategy, the exploring holons are continuously sent out by the order holons, even if a resource has already been reserved. This allows the system to adapt to sudden changes or disturbances in the setup, such as a break down, or to take advantage of an unexpected opportunity. This makes HMES advantageous in terms of robustness.

For this system to work, each holon must contain a model of its own reality. This will enable the holon to predict the outcome of certain scenarios, such as a conveyor section which can predict how long it will take to move a pallet over a certain distance. The researchers tested the HMES on a basic conveyer simulation,

and found that it is suitable to the flow of goods, can handle multiple pallets and can handle disturbances, such as a breakdown or delay in a feeder unit (Van Belle *et al*. 2009).

2.5.3 Cross Docking Strategies

The following discussion is based on research done by Van Belle (2013).

While cross docking problem is not the same as conveyer control, it is possible that cross docking's algorithms could be modified to be used with conveyer systems.

Cross docking is a procedure whereby trucks meet up at a marshalling point and distribute goods to other trucks, through a system of forklifts. The marshalling point consists of number of dock doors where pallets can be loaded onto or removed off trucks. The cross docking problem is to determine which truck must dock at which door, and what route each truck should take to different delivery points to deliver its goods.

Research done into optimising a cross docking procedure, combines the use of holonics and scheduling. The holonic architecture is based on PROSA. It has two scheduling systems, i.e. a Truck Scheduling System (TSS) and a Vehicle Routing Scheduling System (VRSS). The TSS determines which dock doors the trucks must be assigned to, based on an algorithm. The VRSS determines the batching and the routes the trucks must take once they leave the cross dock, also based on an algorithm. Batching determines what pallets each truck should transport. The TSS and the VRSS work with the supervisor holon of the Holonic Logistics Execution System (HLES) to determine the best options for the cross dock.

It is the supervisor holon’s responsibility to provide the VRSS and TSS with the order and resource holons’ information. The VRSS and TSS then update their algorithms accordingly, which the supervisor holon can then access. The supervisor holon then has the responsibility to advise the order and resource holons on what the best solutions are to the cross docking problem.

However, the resource and order holons also have the ability to make use of ant holons. These ant holons are either ‘explorer’ or ‘reservation’ ants. This enables the resource and order holons to learn of problems in the system, such as an unexpected truck delay or forklift breakdown. The holons can then seek the next best solution, and not necessarily follow the scheduling provided by the algorithms. As with the above-mentioned HMES for the conveyor system, this makes the system include one of the strengths of holonics, i.e. that it is robust and able to adapt to unexpected changes in the system.

As mentioned above, cross docking problem is not the same as conveyer control. The conveyer system is less complicated than a cross docking problem, because

there are few, fixed transportation paths (that of the conveyer) while the cross docking problem has many possible permutations due to the number of trucks, forklifts, and paths the trucks and forklifts could take. The concept of a holonic system working with a scheduling algorithm is also an idea which could be used in conveyer control, because the concept includes the strengths of scheduling and holonics (Van Belle 2013).

2.5.4 Airport Baggage Handling System

A common conveyer control problem is that of an airport baggage handling system. Black and Vyatkin (2010) implemented a controller as a basic function block based on IEC61499, which combines the reactive behaviour programmed in ladder logic with higher level functions programmed in Java.

Path planning is done by a central routing controller that has a complete model of the network layout and can perform a tree search of possible paths between points using established path finding algorithms (Black & Vyatkin 2010).

The path planning is done using Dijkstra's algorithm (Cormen *et al*. 2009), which will be explained in further detail in the next section.

2.5.5 Path-Planning Algorithms

While some of the above research only makes use of the ant based traffic control system, a path planning algorithm was used in Black and Vyatkin (2010) and Vrba and Marík (2010). Both these references used Dijkstra's algorithm (Cormen *et al*. 2009).

Dijkstra's algorithm solves the shortest path problem on a graph made up of vertices with connecting edges. The edges can be allocated weights from zero to infinity. A conveyer system can be easily modelled with nodes being designated as vertices and conveyer lengths, or sectors between nodes, being designated as edges. The weight is then the length of an edge between vertices. A computer can then solve the shortest path problem between vertices. If a vertex became unavailable due to physical constraints, such as a break down or if it is occupied by a pallet, then the vertex leading to it would be given the value of infinity. The path finding algorithm would then attempt to find the next shortest path, if another path existed.

While Dijkstra's algorithm can solve the problem of determining the shortest path in a conveyer system, the problem remains of pallet traffic control and determining which paths the pallets must take to their destinations to avoid collisions with each other. Combining Dijkstra's algorithm with flow network analysis may provide a solution both path planning and traffic control. In flow network analysis, the graph edges are labelled with their capacity, and not their

weights. The Ford-Fulkerson algorithm (Cormen *et al*. 2009) can then be used to determine the path which is able to have the maximum capacity of pallets.

3 CONTROL ARCHITECTURE

The section will discuss the project background and physical requirements, leading to a consideration of various possible control architectures. A suitable control architecture which satisfies the requirements is selected and the choice is motivated.

3.1 Typical Application Context

CBI has two factories used for circuit breaker manufacturing. The component factory is in Johannesburg, where the various components of the circuit breakers are manufactured from raw materials. CBI does all the part manufacturing, which gives them the ability to make circuit breakers which cater to a customer's specific needs. Once the parts have been manufactured, the parts are transported by road to the assembly plant in Lesotho.

A critical part of the production process is the quality assurance process, which means testing each circuit breaker to ensure that the circuit breakers work as required. Currently, the quality assurance process involves the workers manually inserting the circuit breaker into a testing machine, which then runs a test on the breaker, and then informs the worker whether the breaker passed the test or not. Failed breakers are sent back to the production line to be reworked, while breakers that passed are sent to the next station.

The problem with the current quality assurance process is that mistakes made by the worker are possible, as it is monotonous, high volume work, which lends itself to errors, such as breakers being placed on incorrect piles, etc. Therefore, the opportunity was there for the MADRG to look at ways at automating processes that will eliminate the human error factor of the quality assurance process.

A part of the human workforce will then be replaced by an automated process. While this will remove some of the job opportunities for the local community, ideally the automated process will speed up the throughput of the production line, and those workers who would have worked on the quality assurance section can be moved to other parts of the production line. It is also of vital importance that CBI can remain competitive with overseas companies, and improving the quality assurance process is part of the ongoing improvement process. While some workers may lose their jobs by being replaced by an automated process, it is better than the company losing its market share due to outdated manufacturing processes. If this were to happen, then substantially more workers would lose their jobs.

The MADRG's focus is, as previously mentioned, on the design of a RQA cell. The current RQA layout is as shown in Figure 5. The figure is very detailed, but the idea is to convey what a full size conveyer system will look like in practice. A

much smaller setup will be used to test the technologies developed in the research thesis.

3.2 Controller Design Requirements

Reconfigurability is a major focus of the project, and it is important that the conveyer (including its control system) must be designed to be reconfigurable. An alternative set-up (which includes physical changes to the conveyor subsystem, corresponding changes to its controller and ramp-up testing) must be achieved with minimal downtime, ideally within a few hours with minimal reliance on specialised technical personnel.

As previously mentioned in Section 1.3, the conveyer consists of various components. The operation of these components would fall under the low level control section of the conveyer controller and is explained in further detail in Section 4.

Apart from the physical operation of the conveyer components, the conveyer controller will also be required to perform high level tasks, such as:

- Communication with the cell controller.
- Communication with other stations in the RQA.
- Communication with the operator through a Human Machine Interface (HMI).
- Path planning.
- Error handling.

These tasks would be grouped as part of high level control. The high level control of the system is explained in further detail in Section 5.

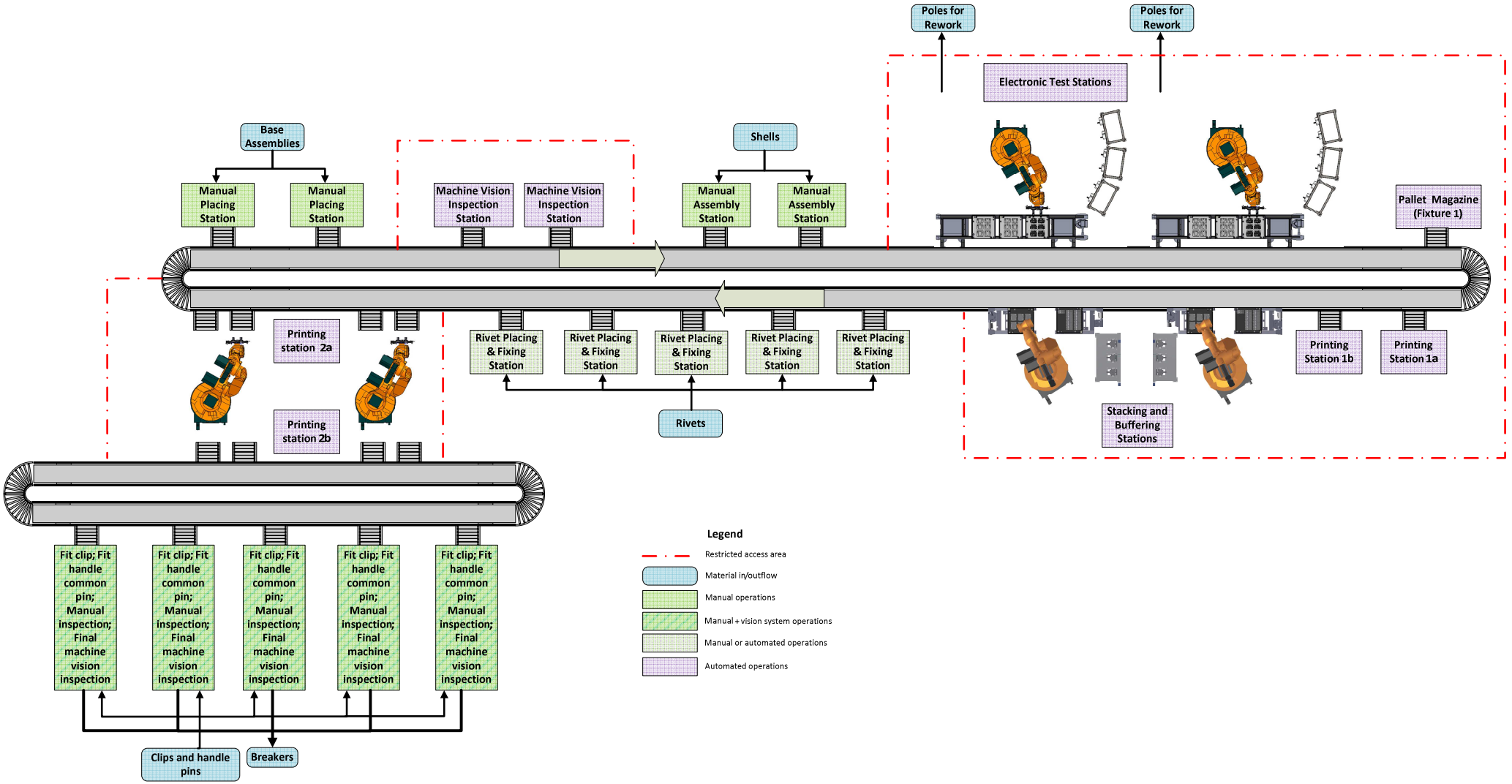


Figure 5: RQA cell layout

3.3 Control Architecture Concepts

The following types and combinations of hardware options were assessed:

3.3.1 Centralised Controller

This option means that one large and powerful controller, typically a PLC, will control the entire conveyer. It will also be responsible for both low level and high level control. It will receive its tasks directly from the cell controller.

- Advantages
  - Very few interfaces are required.
- Disadvantages
  - Does not offer any new benefits to the study of reconfigurability.

3.3.2 HLC Communicating with a Single LLC

This option is will consist of a high level controller (HLC) program written in, for example, Java or C# which will then communicate with a low level controller (LLC) which controls the conveyer hardware. The high level program will handle all communications and high level computations, such as path planning and pallet management.

- Advantages
  - Offers high level programming capability cost effectively.
  - Offers high level communications cost effectively.
  - HLC offers high level communication, such as TCP/IP and XML strings.
- Disadvantages
  - One LLC (typically a PLC) does not offer any benefits to reconfigurability, because control will still be centralised.

3.3.3 HLC Communicating with Multiple LLCs

The HLC will typically run on a PC or industrial PC, and communicate with multiple LLCs which are situated around the conveyer. Each LLC will control a certain section or module of the conveyer. The LLCs will in turn be controlled by the HLC, which will also be responsible for communications with the cell controller and high level computations such as path planning and optimisation.

- Advantages
  - HLC offers high level communication, such as TCP/IP and XML strings.
  - HLC offers high level programming ability.

- Multiple PLCs potentially offer significant reconfigurability benefits.

- Disadvantages
  - Communication between HLC and the LLCs introduces additional interfaces. Additional interfacing hardware may be required.
  - Buying multiple small LLCs could be more expensive than buying one larger PLC.

3.3.4 HLC Communicating with Multiple Remote I/Os

This architecture is similar to that of 3.3.3, except that distributed I/Os are used instead of PLCs. The actuation of the remote I/Os will then be controlled by the HLC.

- Advantages
  - Same strengths as mentioned in 3.3.3.
  - Possibly less expensive than PLCs.
- Disadvantages
  - Same as the weaknesses in 3.3.3.
  - A more advanced HLC program will be required to operate the remote I/Os. Could be beyond the capabilities of a standard PC, and if a PLC is used as the HLC, programming complex logic will be more difficult.

3.4 Selection of Architecture

As stated in Section1.2, an important requirement of the system is reconfigurability. Thus, the options which shall be more closely investigated are that of Sections 3.3.3 and 3.3.4. These options are also a new approach to the reconfigurable conveyer problem, and will offer new research prospects. The centralised controller configurations are well known to have poor reconfigurability, and Le Roux (2013) has already investigated the HLC and LLC configuration described in Section 3.3.2. To confirm that the architecture with multiple LLCs is viable, suitable LLCs were identified.

When the procurement stage of the project was reached, it was required that the PLCs or remote I/O units be procured from a CBI approved supplier. At that stage, the approved supplier was Eaton. CBI and Eaton were also able to offer significant discount on the list price of the PLCs.

After investigation and consultation with the suppliers, it was decided that the Eaton Easy800 PLC would be best suited for the task. This was chosen over the available Eaton remote I/O units because the I/O units did not offer any advantage over the PLCs and, with the required communication module added to each I/O unit, were very similarly priced to the Easy800.

However, the Easy800 was not ideally suited to the requirements. To be able to communicate to a PC via Ethernet, the Easy800 requires an Ethernet gateway adaptor, which is more expensive than the Easy800 itself. Communication between Easy800s is possible through network cables plugged into built-in ports. Thus, to cut costs, the hardware architecture would have been to have one Easy800 with the Ethernet gateway acting as the 'server' PLC. All the Easy800s would have been connected in series with each other and connected to the server PLC through this method. While not ideal, this set-up should have worked. The setup is shown graphically in Figure 6. Note that that dotted box encapsulates the transport section of the cell. The cell controller is not part of the project; it only needs to be communicated with.

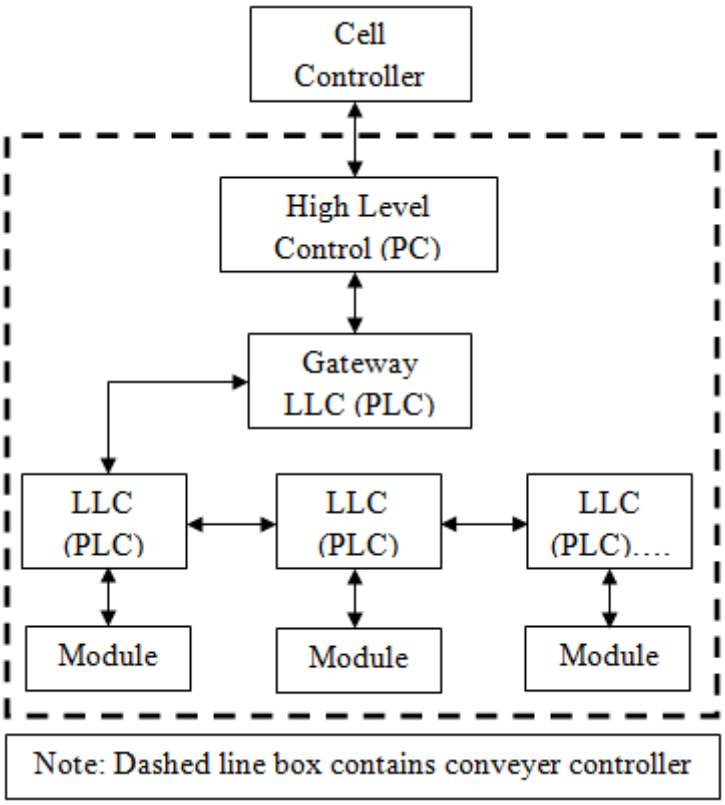


Figure 6: Eaton PLC control architecture

CBI then unexpectedly changed their supplier requirements to that of LS Industrial Systems (LSIS). Again, the option of using distributed I/Os was investigated, however the price difference was small enough that it was thought to be more beneficial to the project if PLCs were used.

The PLC which best suited the project's requirements was the XEC series of PLCs. They are available in different I/O configurations, and the one deemed most suitable for the project was the version with 8 relay outputs and 12 inputs (XEC-DR20SU). This version can also be expanded to handle more I/Os if

necessary and can be linked to a PC via mini-USB for programming. All programming software comes with the PLCs.

An advantage of the LSIS PLCs is that the LSIS Ethernet gateway adaptor is relatively low cost, and it is therefore viable to buy one gateway for each PLC. This means that each PLC is independently connected to the HLC via Ethernet. The hardware architecture is then precisely as described in Section 3.3.3. The structure is shown graphically shown in Figure 7.

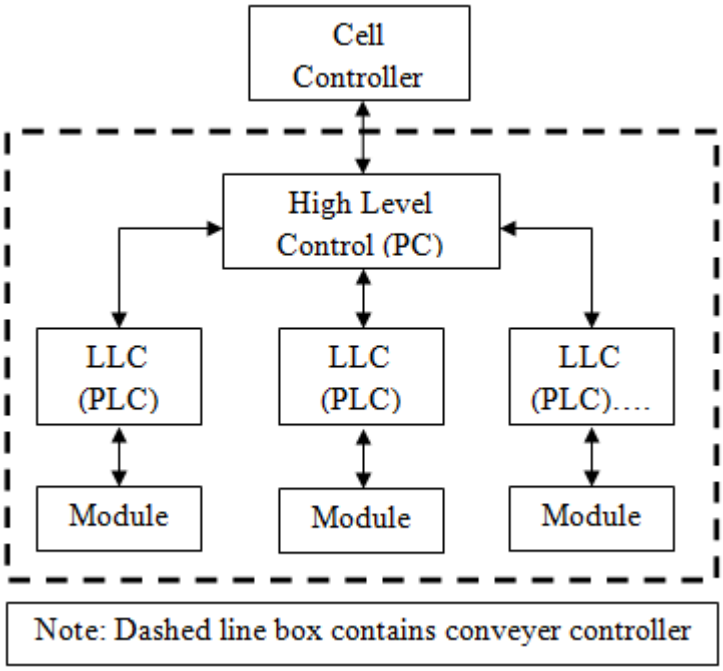


Figure 7: Final control structure

Another supplier that was investigated, when CBI changed vendors, was Siemens. Siemens offers the LOGO range of PLCs, which is similar to the XEC-DR20SU, except that Ethernet communication is built into each unit. This could have offered a substantially cheaper solution, and possibly an easier solution as well, because most of the Automation Laboratory's equipment uses Siemens PLCs already. However, CBI required that their products be used, therefore the XEC-DR20SU units were chosen.

A few months after the XEC-DR20SUs had been procured, LSIS brought out another type of PLC, the XEC-DN32U. The difference between this and the DR20SU is that it has two inbuilt Ethernet ports, and more I/0s. However, the DN32U units would still have been more expensive than the DR20SU units with their Ethernet modules. A price comparison of all products looked at is given in Table 1.

Because the university is an academic institute, discount prices were offered on all products, which are included in the table. All prices are in South African Rands in March 2015.

Table 1: PLC price comparison

Make	Model Name	List Price	Discount Price	No.	List Total	Discount Total
LSIS	XEC-DR20SU PLC	3965	1627	7	27758	11390
	XBL-EMTA Ethernet Module	4250	1827	7	29754	12790
				Total	**57512**	**24180**
LSIS	XEC-DN32U PLC	10256	7179	7	71789	50252
				Total	**71789**	**50252**
Eaton	EASY821-DC-TC PLC	6500	3250	7	45500	22750
	EASY-NT-R Port Plugs	280	140	2	560	280
	EASY800-USB-CAB Cable	2300	1150	1	2300	1150
				Total	**48360**	**24180**
Siemens	6ED1052-1CC01-0BA8 PLC	1490	894	7	10430	6258
	6ED1052-0BA8-0YA1 Software	654	654	1	654	654
				Total	**11084**	**6912**

.

4 LOW LEVEL CONTROL

This section discusses how the conveyer system was separated in modules, and describes the function and operation of each module in detail. The various components that make up the modules are also discussed.

4.1 Identification of Modules

Because the focus of the project, as stated in Section 1.2, is a modular controller for the conveyer, the conveyer needs to be separated into logical modules. Each module is required to perform a specific task. All components required to perform that specific task is grouped into a module. A module is then controlled by its own LLC, implemented on a LSIS PLC, as motivated in Section 3.4.

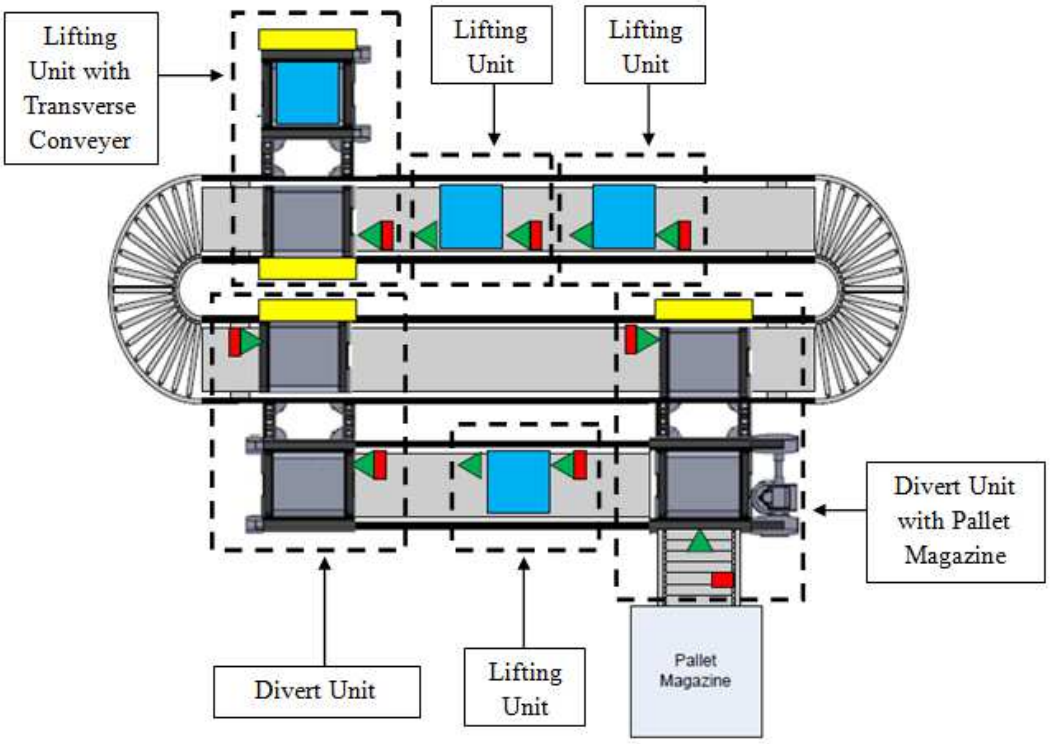


Figure 8: Distribution of conveyer modules

Figure 8 shows the conveyer setup as it is in the Automation Laboratory. This is a simplified version of the setup shown in Figure 5, consisting of just a modified section of it. Because of size and equipment constraints, the setup shown in Figure 5 could not be reproduced in the Automation Laboratory.

The separate modules have been labelled and identified with black dotted line boxes. These modules are consistent with that which would be used in the larger setup shown in Figure 5. The types of modules are:

- Lifting Unit.
- Lifting Unit with Transverse Conveyer.
- Divert Unit.
- Divert Unit with Pallet Magazine.

Each individual module is explained in more detail from Section 4.4 onwards.

4.2 LLC Hardware Components

This section describes the components used to create the modules. Except for the PLC, all the hardware described in this section is standard TS2 Plus components.

4.2.1 PLC

As mentioned, each module is made up of components. The central component is the controller, the LSIS PLC.

Figure 9: LSIS PLC

Figure 9 shows the LSIS PLC as part of a lifting unit module. All PLCs are wired and housed in a similar way.

The XEC PLC units can be programmed using any computer which has the LSIS software installed. Initial programming must be done through a USB connection, but once the network properties of the unit have been configured, the units can be communicated with through standard Ethernet and network connections. This made the programming and debugging of the system much easier when the PLCs were installed on the conveyer.

The PLCs can be programmed using ladder logic, sequential charts or structured text formats.

4.2.2 Lifting Component

The lifting component (Figure 10) can be fitted with a conveyer track and is pneumatically operated with two position settings. The low setting lets a pallet pass along the conveyer track without hindrance. The high setting lifts a pallet (that has been stopped correctly above it) slightly off a conveyer and holds it steady, so that work can take place on the parts that the pallet is carrying.

4.2.3 Stop Gate

The stop gate (Figure 11) is fitted on the inside of a conveyer track and is also pneumatically actuated. On a signal from the controller, the stop gate will either lift or lower its gate, thereby stopping or letting a pallet through. The stop gates are unidirectional; they can only stop a pallet moving in one direction. Stop gates are used to control the flow of pallets and to stop pallets in the correct position above lifting components.

4.2.4 Proximity Switch

The proximity switch generates a signal when a metallic material passes over it. All pallets have metallic strips on their undersides, which trip the proximity switch as the pallet passes over them. In Figure 10, the proximity switches have been combined with the stop gates to form one unit.

4.2.5 Rocker Proximity Switch

The rocker proximity (Figure 13) switch has a rocking barrier which in its free position is inclined towards the conveyer. It also has a metal strip, similar to that of the pallet. When a pallet pushes against the barrier, the barrier “rocks” back, and trips a proximity switch. Note that Figure 13 is not placed here since it will be considered in detail in Section 4.4.2.

4.2.6 Transverse Conveyer

The transverse conveyer (Figure 13) is a pneumatically actuated component which is used to move a pallet off the conveyer. It has three possible settings: The low-

level setting allows pallets to pass unhindered. The mid-level setting will stop a pallet precisely above the transverse conveyer, in position ready to be transferred off the conveyer path. The mid-level setting is the default setting for this component. The high-level setting lifts the pallet clear of the conveyer, and when the transverse conveyer's motors are switched on, the pallet is transferred. The motors can be unidirectional or bidirectional, depending on the function of the module. The transverse conveyer can also be seen Figure, which is not placed here since it will be considered in more detail in Section 4.4.3.

4.3 Common Aspects

The following section discusses the common aspects of the modules.

4.3.1 Programming Structure

The PLC's programming is based on CASE statements. The Structured Text programming format was therefore chosen to facilitate this. The default state of all PLCs is Case 0. In this state, all stopgates are down and the PLC allows pallets to pass without stopping them. The PLC is not able to accurately detect the passing of pallets in this state because the pallets pass over the proximity switches too quickly. In this state, the module is non-operational and effectively an open path.

Case 1 is the "Stop and Check" state. This is the state a PLC enters when it is active. The stopgates are in the up position and the PLC stops pallets as they reach the module. This allows the proximity sensors to detect that a pallet is in the vicinity of the module, and the HLC program can make a decision and send further instructions to the PLC if required. The other case states of the program depend on the module's tasks and functions. The logic flow of each module will be explained in Section4.4.

The lifting unit module and divert unit with pallet magazine module's PLC programs are shown in Appendix E, to illustrate the programming structure.

4.3.2 Communication

As mentioned in Section 3.4, the PLCs selected for the LLCs can be fitted with Ethernet adapters. This enables the LLCs to communicate using the TCP/IP protocol. The PLCs are programmed to have both a server and client channel. After start-up, the server channel continuously listens for a client trying to connect to it, and the client channel continuously tries to connect to a server whose port and IP address matches that specified in the communications setup. Once the server and clients have successfully connected, data transmission can take place.

Data sent by the HLC and received by the PLC is stored in a specific section of memory in the PLC. The PLC uses this data to execute a serious of commands contained in a CASE statement. The PLC is programmed to send a byte of data to the HLC every 200ms. The four least significant bits contain the state the PLC is currently in, and the next four bits are linked to the PLC's sensors. Based on the

bytes the HLC receives every 200ms, the HLC sends the CASE state number that the HLC wants the PLC to execute to the PLC. The maximum amount of inputs the PLC will require is three, and the maximum amount of states a PLC will require is seven, therefore, one byte is more than adequate to convey the data required. In the case that more data is required than what one byte can convey, the TCP/IP settings can easily be modified to send more bytes of data.

4.4 Detailed Explanation of Modules

This section discusses the functions, interfaces and implementation of each module. An annotated photograph and diagram is provided to show what each module looks like and where each component fits in. Each module also has a table showing the various I/Os, or interfaces, each module PLC has. Finally, a state diagram is given to show the various functions a module can execute, and the order in which they can take place. Each function block contains a four digit number, which denotes the CASE state number the module is currently in. Note that the numbering format is in binary. More detailed state diagrams for each module are included in Appendix D.

A description of the symbols used is shown in each of the module diagrams are shown in Table 2.

Table 2: Module diagram symbol allocation

Symbol	Description
Blue Square	Lifting Component
Green Triangle	Stop Gate
Red Rectangle	Proximity Sensor
Yellow Rectangle	Rocker Proximity Switch

Note that the numbered labels in the module diagrams correspond to the numbering in the I/O table provided for each module.

Also note that all I/O tables show a “Main Motor Line” connection. All module PLCs have one output connected to a central bus, which is connected to the motors which operate the main conveyer track. If any one of the PLCs activates that output, the conveyer track is activated and will move any pallets on the conveyer. The only time this output is not active is when a module is in the default (CASE 0) state.

4.4.1 Lifting Unit

The lifting unit module is the simplest of the modules which make up the conveyer. Its task is to lift a pallet off the conveyer and hold it steady while work is done on the parts being carried by the pallet.

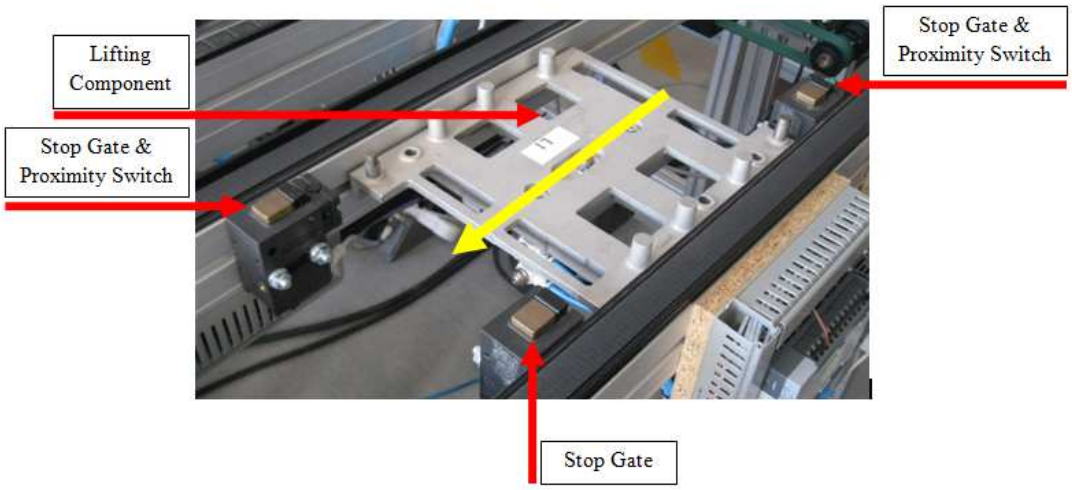


Figure 10: Lifting unit component photograph

Figure 10 shows a photograph of the lifting unit module. The lifting component, stop gate and proximity switches can also be seen here. Note that the stop gates and proximity switches have been combined into one compact unit.

When Figure 10 is compared to Figure 11, an extra stop gate and proximity switch unit can be seen in Figure 10. This unit is part of another module further along the conveyer belt.

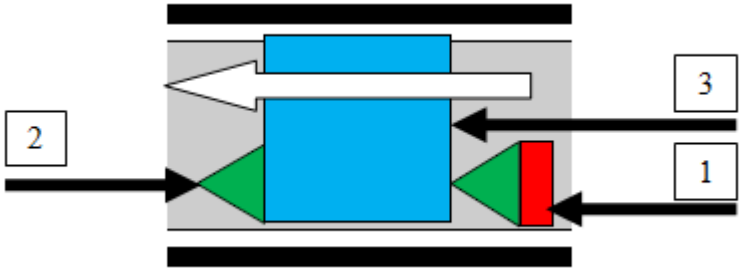


Figure 11: Lifting unit diagram

Figure 11 shows the diagram of a lifting unit module. The stop gate and proximity switch positioned before the lifting component are there to stop a pallet, and to signal to the controller that a pallet is in proximity. The stop gate positioned after the lifting component is there to stop a pallet in the correct position, so that when the lifting component is actuated, the pallet will be successfully lifted and held in position.

The inputs and outputs that will be required by a controller to operate the module's hardware are shown in Table 3:

Table 3: Lifting unit I/O summary

	Hardware	Inputs	Outputs	Pneumatic
1	Stop Gate/Proximity Switch A	1	1	Yes
2	Stop Gate B		1	Yes
3	Lifting Component		2	Yes
4	Motor Main Line		1	
	Total	**1**	**4**	

Figure 12 shoes the lifting unit module state diagram. From the Stop and Check state, the module can go into two possible states: It can either let the pallet pass through it, or can initiate the Lift Pallet sequence which lifts the pallet off the conveyer.

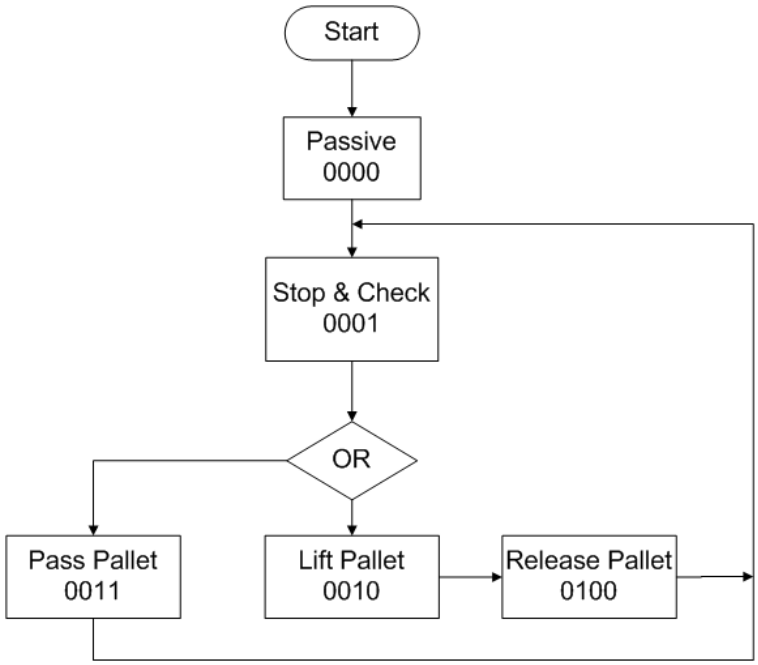


Figure 12: Lifting unit flow diagram

The details of each state are:

- Passive
  - Both stop gates and the lifting component are in the low position. Pallets are allowed to pass unhindered. The module is effectively an open path.

- Stop and Check.
  - Stop Gate A is set to the high position. Pallets are stopped above the proximity switch so that the module is made aware of the pallets position. Stop Gate B and the lifting unit are in the low position.
- Pass Pallet
  - Both stop gates are lowered, to allow the pallet through.
  - After a period of time determined by a timer in the PLC program, Stop Gate A is lifted again to stop the next pallet coming through.
  - The module returns to the Stop and Check state.
- Lift Pallet
  - Stop Gate A is lowered and Stop Gate B is kept high.
  - After a short period of time to wait for the pallet to move over Stop Gate A, Stop Gate A is raised to prevent the next pallet from interfering.
  - After giving the pallet enough time to move into position above the lifting component, the lifting component is actuated to its high setting to hold the pallet in position. Note that Stop Gate B stops the pallet in the correct position above the lifting component.
  - The module remains in this state until it is told to change to the Release Pallet state.
- Release Pallet
  - The lifting component is actuated to the low position, and Stop Gate B is lowered.
  - After giving the pallet some time to move away from the module, the module returns to the Stop and Check state.

Note that the lifting unit module cannot allow a pallet to pass if it is occupied by another pallet. It will cause a blockage in the path until the occupying pallet has been released.

The lifting unit module and lifting unit with transverse conveyer module (Section 4.4.2) are destination modules, which means that they can hold pallets. The other modules are purely there to divert the pallet to another part of the conveyer.

4.4.2 Lifting Unit with Transverse Conveyer

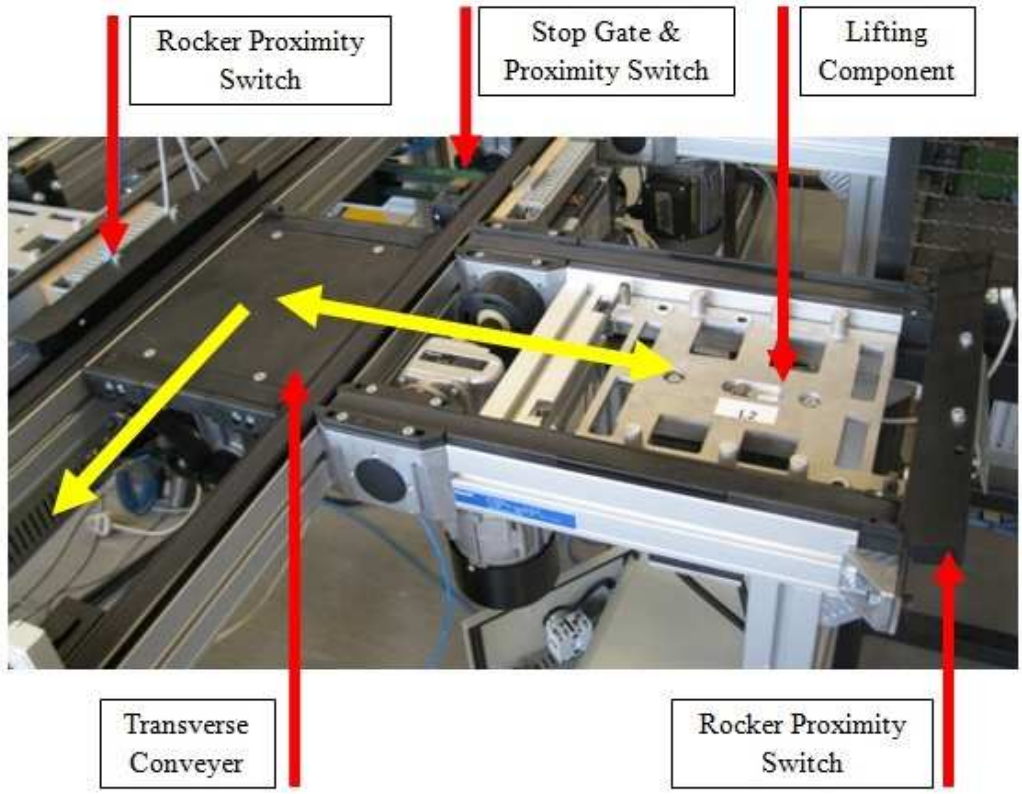


Figure 13: Lifting unit with transverse conveyer photograph

Figure 13 shows the annotated photograph of the lifting unit with transverse conveyer module. This module's task is to transfer a pallet off the main track, and then to use a lifting component to hold the pallet steady. In addition to a stop gate, proximity switch and a lifting component used in the lifting unit module, this module also makes use of a rocker proximity switch and a transverse conveyer. Note that this module has the ability to pass a pallet even when the lifting component is occupied.

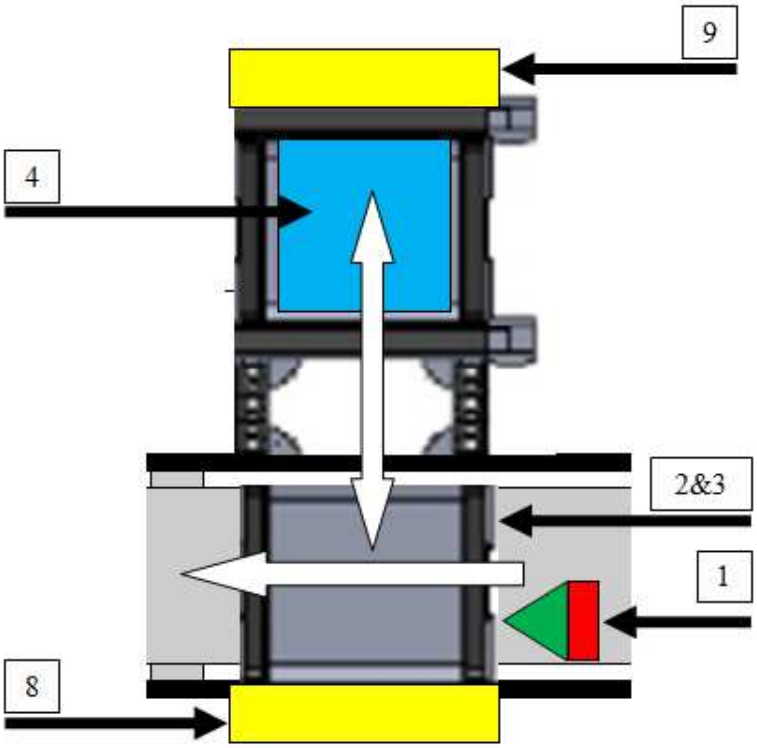


Figure 14: Lifting unit with transverse conveyer diagram

The stop gate and proximity switch unit shown in Figure 14 is positioned before the transverse conveyer to stop pallets and to send a signal to the controller when a pallet is in proximity. The two rocker proximity switches are used to determine when the pallet has reached the ends of the transverse conveyer.

Note that the transverse conveyer's motor are contactor actuated, and are bidirectional for moving the pallet either towards the lifting component, or to move it back to the conveyer.

The inputs and outputs required by the controller to operate the lifting unit with transverse conveyer module are shown in Table 4.

Table 4: Lifting unit with transverse conveyer I/O summary

	Hardware	Inputs	Outputs	Pneumatic
1	Stop Gate/Proximity Switch	1	1	Yes
2	Low-level Transverse Conveyer Setting		1	Yes
3	High-level Conveyer Setting		1	Yes
4	Lifting Component		1	Yes
5	Motor Direction Control A		1	
6	Motor Direction Control B		1	
7	Motor Main Line		1	
8	Rocker Proximity Switch A	1		
9	Rocker Proximity Switch B	1		
	Total	**3**	**7**	

Figure 15 shows the flow diagram for the lifting unit with transverse conveyer module. After the Stop and Check state, the module can wither execute the Pass Pallet case, Divert and Lift case or the Release Pallet Case.

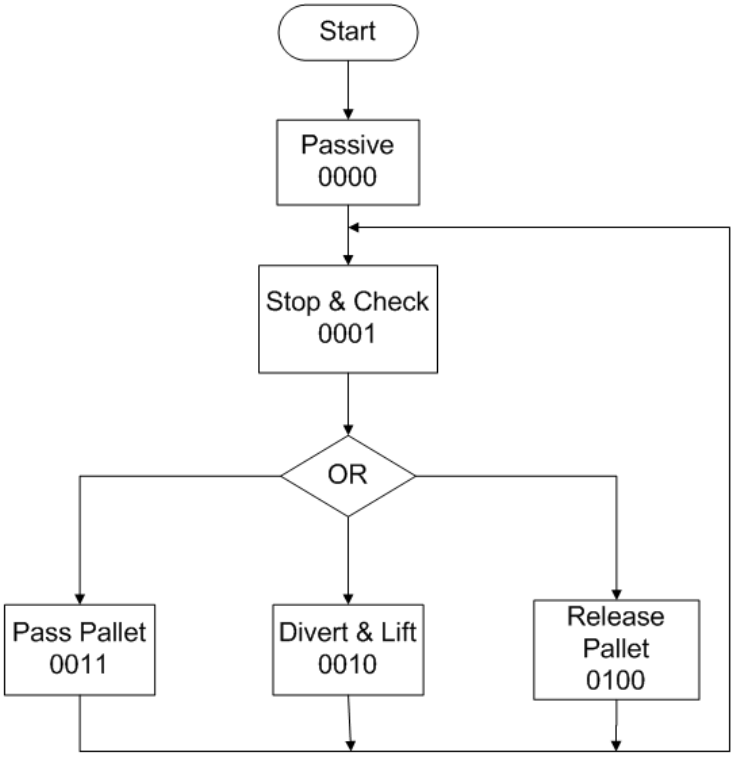


Figure 15: Lifting unit with transverse conveyer flow diagram

The particulars of each state are:

- Passive
  - The stop gate is in the low position, and the transverse conveyer is on its low-level setting. Pallets are allowed to pass unhindered. The module is effectively an open path.
- Stop and Check.
  - The stop gate is set to the high position. Pallets are stopped above the proximity switch so that the module is made aware of the pallets position. The transverse conveyer is still in the low-level setting.
- Pass Pallet
  - The stop gate is lowered, and the transverse conveyer is actuated to low level.
  - After a certain amount of time to allow the pallet to pass over the transverse conveyer, determined by a timer in the PLC's program, the stop gate is raised again.
  - The module returns to the Stop and Check state.
- Divert & Lift Pallet
  - The stop gate is lowered, and the transverse conveyer is actuated to mid-level setting, in position to stop a pallet above it.
  - After a certain amount of time to allow the pallet to pass, determined by a timer in the PLC's program, the stop gate is raised, and the transverse conveyer is raised to high-level setting. The motors are activated to move the conveyer towards the lifting component.
  - When the conveyer makes contact with Rocker Proximity Switch B, the motors stop, and the lifting component is actuated to hold the pallet in position.
  - The module returns to the Stop and Check state.
- Release Pallet
  - The stop gate is kept up to prevent interference from other pallets. The lifting component is lowered. The transverse conveyer is actuated to high, and the motor is activated to move the pallet towards the main conveyer track.
  - When the pallet makes contact with Rocker Proximity Switch A, the motor is deactivated, and the transverse conveyer is actuated to the low level setting, to allow the pallet to move away from the module.
  - The module returns to the Stop and Check state.

4.4.3 Divert Unit

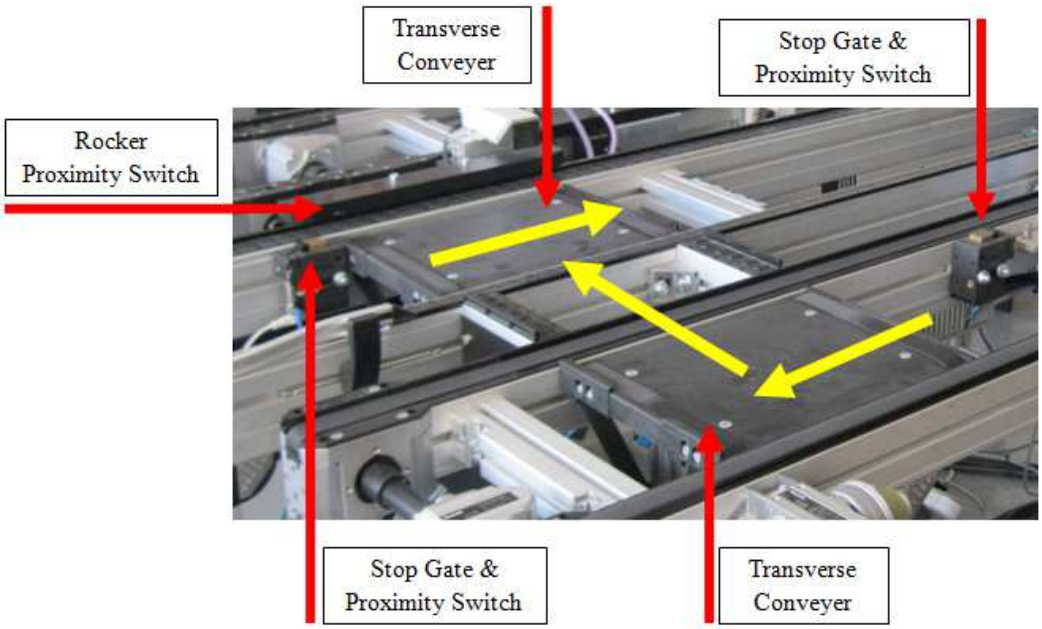


Figure 16: Divert unit module photograph

Figure 16 shows the divert unit module. This module transfers a pallet from one conveyer track to another parallel conveyer track. It also makes use of transverse conveyers. Note that this is a unidirectional transverse conveyer; it can only transfer pallets in one direction.

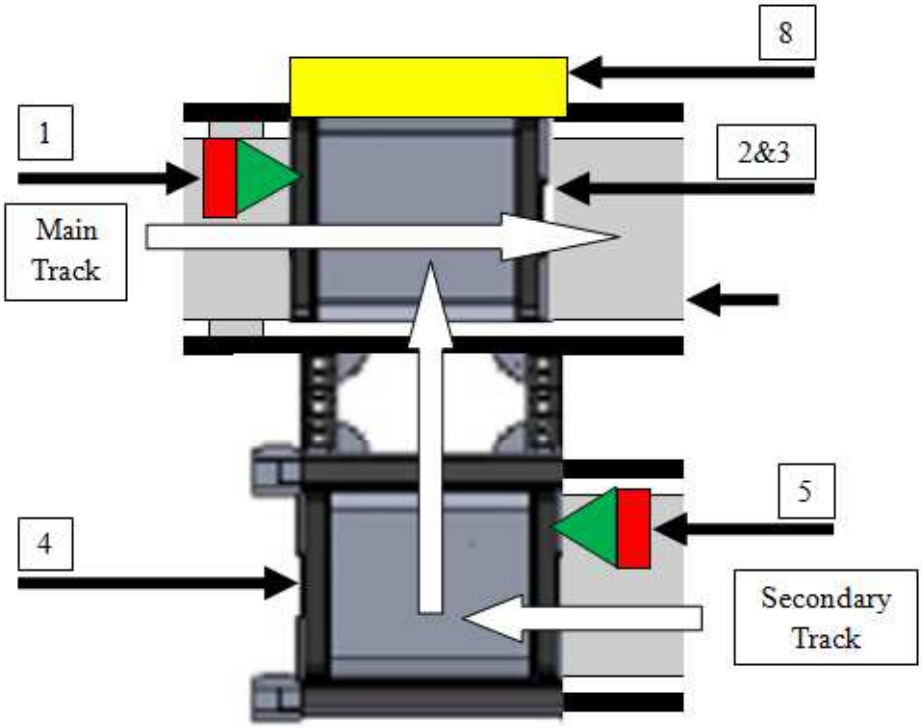


Figure 17: Divert unit diagram

Figure 17 shows the diagram of the divert module. Note that the main and secondary conveyer tracks are labelled next to the direction flow arrows. Stop Gate and Proximity Switch A are there to stop pallets on the main track and to notify the LLC of pallet proximity. Stop Gate and Proximity Switch B are there to perform the same task on the secondary track. Transverse Conveyer A and B enable the pallets to be moved from the secondary track to the main track. The rocker proximity switch notifies the LLC when a pallet has moved to the edge of Transverse Conveyer A.

The inputs and outputs required by the controller to operate the divert unit module are shown in Table 5.

Table 5: Divert unit module I/O summary

	Hardware	Inputs	Outputs	Pneumatic
1	Stop Gate/Proximity Switch A	1	1	Yes
2	Transverse Conveyer A (low position)		1	Yes
3	Transverse Conveyer A (high position)		1	Yes
4	Transverse Conveyer B (high position)		1	Yes
5	Stop Gate/Proximity Switch B	1	1	Yes
6	Motor On		1	
7	Motor Main Line		1	
8	Rocker Proximity Switch A	1		
	Total	**3**	**7**	

Figure 18 shows the flow diagram for the divert unit module. There are only two possible states, after Stop and Check, Pass Pallet on the main track and Divert Pallet from thesecondary track to the main conveyer track.

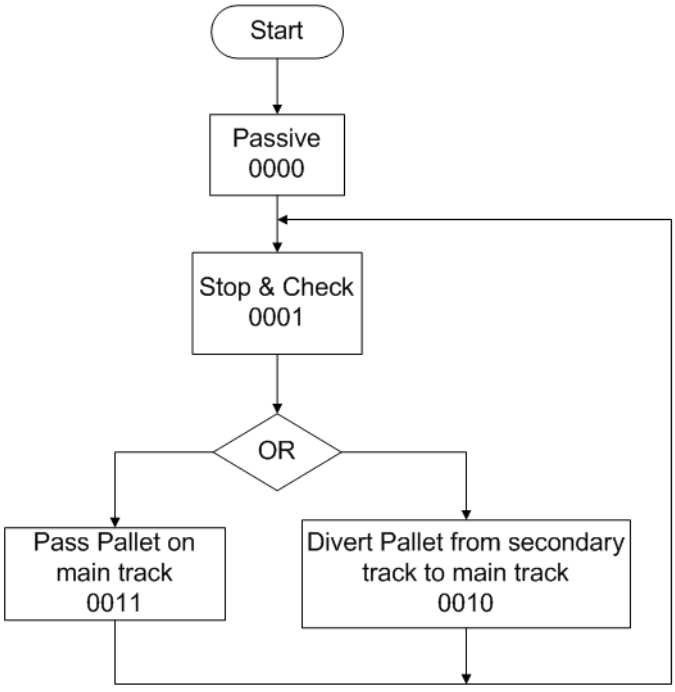


Figure 18: Divert unit flow diagram

- Passive
  - Stop Gate A is in the low position. It is effectively an open path. Stop Gate B is in the high position. Because there is no track beyond the module, it is not hampering the passage of any pallets.

Stop Gate B is kept high to prevent pallets from moving into position above Transverse Conveyer B. Transverse Conveyer A is in the low position and Transverse Conveyer B is in the mid-level position.

- Stop and Check
  - Stop Gate A is in the high position. Pallets are stopped above the proximity switch so that the module knows when a pallet has arrived. All other components are in the same position as that in the Passive State.
- Pass Pallet
  - Stop Gate A is lowered and Transverse Conveyer A is actuated to low level position.
  - After a period of time for the pallet to pass over Stop Gate A, Stop Gate A is raised again.
  - The module returns to Stop and Check state.
- Divert Pallet
  - Stop Gate A is kept high, Stop Gate B is lowered to allow the pallet to move onto Transverse Conveyer B. Transverse Conveyer B is in its mid level position to stop the pallet directly above it.
  - After a period of time to allow the pallet to move into position, Transverse Conveyers A and B are actuated to the high position. Stop Gate B is set to high to stop other pallets on the secondary track. The motor is activated to move the pallet towards the main conveyer track.
  - When the pallet trips the rocker proximity switch, Transverse Conveyer A is actuated to the low level setting allow the pallet to move away from the module.
  - The module returns to the Stop and Check state.

Note that this module has more than one entry point. In the scenario where a pallet trips the proximity switch on both entry points, priority is given to the secondary track, because it has a smaller capacity for pallets, therefore a larger chance of congestion.

4.4.4 Divert Unit with Pallet Magazine

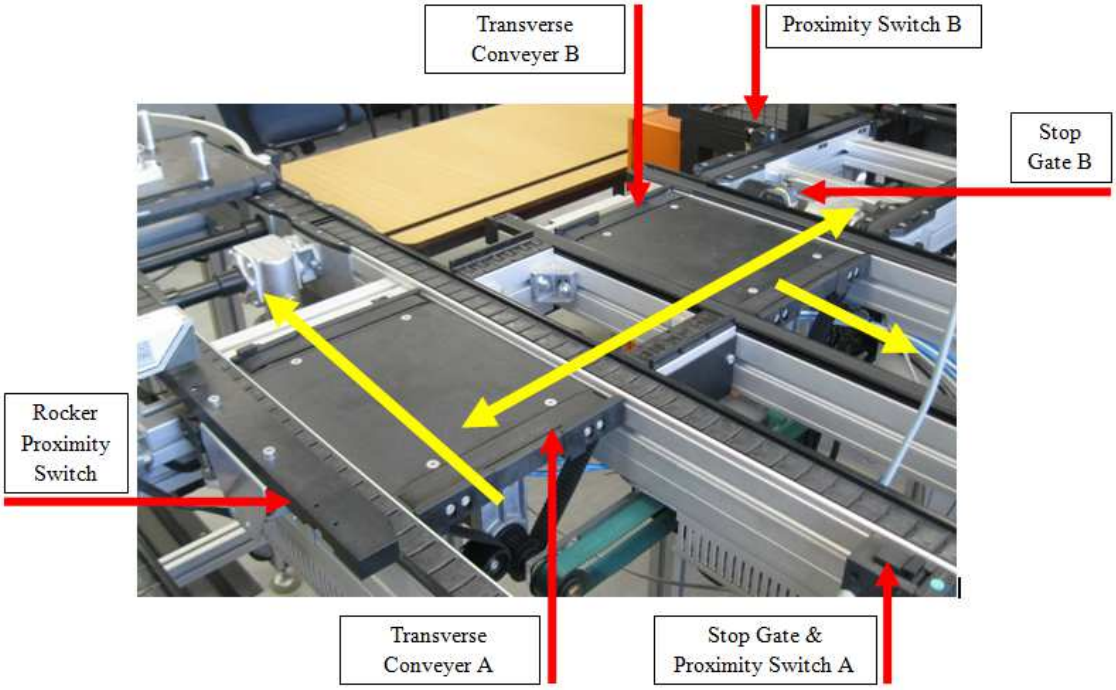


Figure 19: Divert unit with pallet magazine photograph

Figure 19 shows the divert unit with pallet magazine module. This is the most complicated module. Its functions are to transfer pallets between the main conveyer track, secondary conveyer track and the pallet magazine.

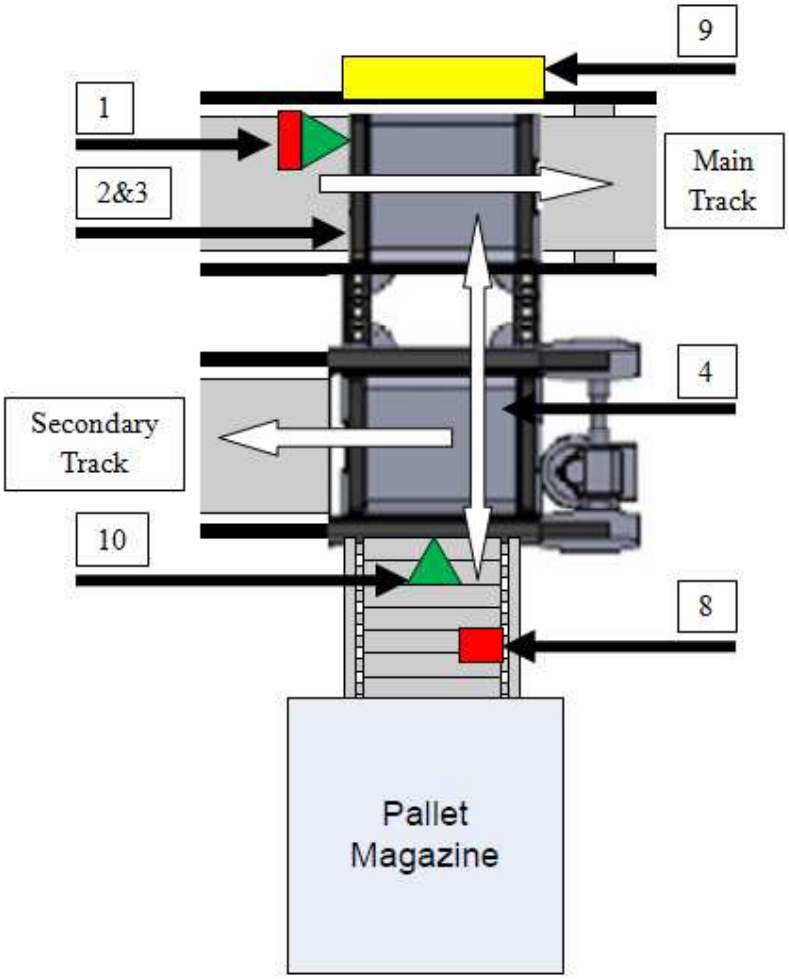


Figure 20: Divert unit with pallet magazine diagram

Figure 20 shows the diagram of the divert unit with pallet magazine module. The conveyer path with the rocker proximity switch and Stop Gate/Proximity Switch A is on the main conveyer track. Stop Gate/Proximity Switch A is to stop and register pallets on the main track. Stop Gate B is different from the other stop gates in that is can be used to stop pallets going in both directions. When a pallet moves from the main track to the secondary track, Stop Gate B stops the pallet from moving too far. It is there to stop the pallet exactly above the transverse conveyer, which can then lower and allow the pallet to move off the transverse conveyer.

The inputs and outputs required by the controller to operate the divert unit with pallet magazine module are shown in Table 4.

Table 6: Divert unit with pallet magazine I/O summary

	Hardware	Inputs	Outputs	Pneumatic
1	Stop Gate/Proximity Switch A	1	1	Yes
2	Low-level Transverse Conveyer Setting A		1	Yes
3	High-level Conveyer Setting A		1	Yes
4	High-level Conveyer Setting B		1	Yes
5	Motor Direction Control A		1	
6	Motor Direction Control B		1	
7	Motor Main Line		1	
8	Proximity Switch B	1		
9	Rocker Proximity Switch	1		
10	Stop Gate B		1	
	Total	**3**	**8**	

The logic for the divert unit with pallet magazine module was the most complex. In this module, there are two input and three output track options. Each input track has a sequence which enables a pallet to be transferred to any of the output tracks. A complication with this module is that the input track coming from the pallet magazine could also serve as an output track.

Priority was given to the input track which leads from the pallet magazine, because it has no buffer, therefore any pallet sensed on that track needs to be moved immediately. Because of the multiple input and output track options, this module has seven possible CASE states in can be in. Below is the flow diagram for the states.

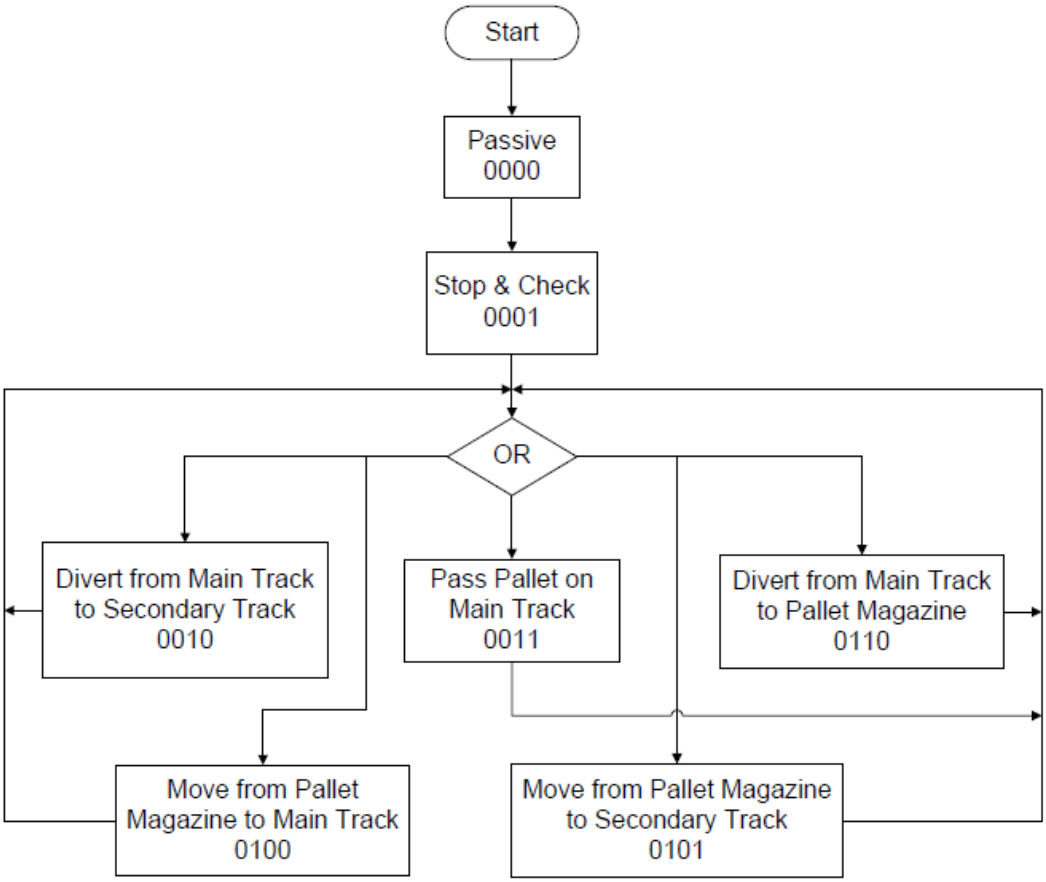


Figure 21: Divert unit with pallet magazine flow diagram

- Passive
  - Stop Gate A is in the low position. The module is effectively an open path on the main track. Stop Gate B is in the high position. This is to prevent pallets which have been unintentionally or launched too early from the pallet magazine from going past Proximity Switch B without being registered. All the transverse conveyers are in the low position which allows pallets to pass unhindered.
- Stop and Check
  - Stop Gate A is in the high position. All other components are the same as what they were in the Passive state. Pallets on the main conveyer track are now stopped in position above Proximity Switch A by Stop Gate A, and pallets coming from the pallet magazine are stopped in position above Proximity Switch B by Stop Gate B.
- Divert from Main Track to Secondary Track

- Stop Gate A is lowered and Transverse Conveyer A is actuated to the mid level position to stop the pallet above Transverse Conveyer A.
- After a period of time to allow the pallet to move over Stop Gate A, Stop Gate A is raised to prevent oncoming pallets from interfering.
- After a period of time to allow the pallet to move in position above Transverse Conveyer A, Transverse Conveyer A and Transverse Conveyer B is actuated to the high level position. Stop Gate B is in the high position. The transverse conveyer motor is activated, to move the pallet towards Stop Gate B.
- After a period of time has passed to allow the pallet to make contact with Stop Gate B and be stopped in position above Transverse Conveyer B, the motor is stopped. Transverse Conveyer B is actuated to its mid level position to allow the pallet to move off Transverse Conveyer B and away from the module.
- The module returns to the Stop and Check position.

- Pass Pallet on Main Track
  - Stop Gate A is lowered. Transverse Conveyer A is set to low level position,
  - After a period of time to allow the pallet to move over Stop Gate A and Transverse Conveyer A, Stop Gate A is lifted to stop other pallets.
  - The module returns to the Stop and Check state.
- Divert from Main Track to Pallet Magazine
  - Stop Gate A is lowered and Transverse Conveyer A is actuated to the mid level position to stop the pallet above Transverse Conveyer A.
  - After a period of time to allow the pallet to move over Stop Gate A, Stop Gate A is raised to prevent oncoming pallets from interfering.
  - After a period of time to allow the pallet to move in position above Transverse Conveyer A, Transverse Conveyer A and Transverse Conveyer B is actuated to the high level position. Stop Gate B is set low. The transverse conveyer motor is activated, to move the pallet towards the pallet magazine.
  - When Proximity Switch B is tripped, the pallet is in position to be loaded into the pallet magazine. The transverse conveyer motor is stopped.
  - The module returns to the Stop and Check state.
- Move from Pallet Magazine to Main Track

- Transverse Conveyer A and B are set to high level position. Stop Gate B is set low. The transverse conveyer motor is activated, with direction being to move the pallet from the pallet magazine towards the conveyer main track.
- When Rocker Proximity Switch A is tripped by the pallet, Transverse Conveyer A is lowered to allow the pallet to move away from the module.
- The module returns to the Stop and Check position.

- Move from Pallet Magazine to Secondary Track
  - Transverse Conveyer A and B are set to high level position. Stop Gate B is set low. The transverse conveyer motor is activated, with direction being to move the pallet from the pallet magazine towards the conveyer main track.
  - When Rocker Proximity Switch A is tripped by the pallet, the motor direction is reversed, to move the pallet towards the secondary track. Stop Gate B is raised to stop the pallet above Transverse Conveyer B.
  - After a period of time has passed to allow the pallet to make contact with Stop Gate B and be stopped in position above Transverse Conveyer B, the motor is stopped. Transverse Conveyer B is actuated to its mid level position to allow the pallet to move away from the module.
  - The module returns to the Stop and Check position.

The reason why the pallet must first move to Transverse Conveyer A and reverse its direction is because there is no stop gate installed to stop the pallet above Transverse Conveyer B, if the pallet is coming in from the pallet magazine.

4.5 LLC input and output summary

The total amount of outputs required by the distributed LLC is 33, and the total amount of inputs required by the LLC is 12. When the conveyer control system is divided up into separately controlled modules and controlled separately, the maximum number of I/Os a low level controller will require is seven outputs and three inputs. This is within the capabilities of the PLC chosen, which has 8 outputs and 12 inputs.

5 HIGH LEVEL CONTROL

This section discusses the high level control of the conveyer controller. The requirements and chosen architecture are discussed. The reasons behind the selection of the implementation platform are discussed and compared to other platforms. The holons which make up the architecture are described in detail, as well as the data structures required by the holons.

5.1 Requirements

As mentioned in Section 1.2, the main focus in this project is reconfigurability and therefore, the software architecture must conform to the characteristics of reconfigurability, which were stated in Section 2.1. It must also perform the high level functions stated in Section **Error! Reference source not found.**.

5.2 HLC Architecture

As stated in the literature review, holonic control has significant benefits in achieving reconfigurability. With this in mind, a holonic architecture was used for the control program.

The two generally accepted holonic architectures are PROSA and ADACOR. Because the objective is to eventually optimise the transportation process, it was felt that the ADACOR architecture, with its central supervisor holon, would be more suitable to the task. An architecture such as that shown in Figure 3 was chosen. However, modifications and additions were made to the traditional ADACOR architecture to make it more suitable for the conveyer system. The types of holons used and their abbreviations are:

- Conveyer Controller Holon (ConC)
- Conveyer Map Holon (ConMap)
- Pallet Holon (PH)
- Operational Holon (OH)

Figure 22 shows the holon hierarchy and communication between the holons. Each holon and the way they interact with each other are explained in detail in Section 5.5.

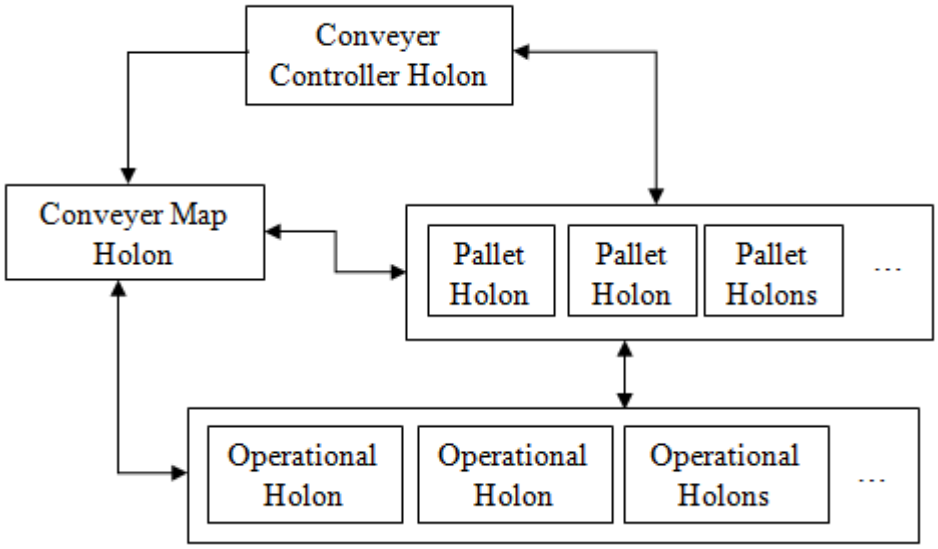


Figure 22: Holon communication and hierarchy

5.3 Implementation Platform

The programming was done in the C# programming language. Microsoft products are ubiquitous in the computing world; therefore C# is widely accepted. It also has widespread use and acceptance in industry. It is also part of the ongoing research in the MADRG to determine the suitability of C# to holonic applications.

C# characteristics of encapsulation and inheritance also have significant advantages for a holonic programming structure. A base class of holon can be created, and the more specialised holons can then inherit from the base class. This saves on repetitive programming, and encapsulates functions in one class, also making debugging and creating new holons easier. C# also has a well-developed graphical user interface functions, making it easy to create HMIs. The Microsoft Development Network website offers good guidance on how to use the program, and is always available on the internet. Michaelis & Lippert (2013) was also used as a reference.

One of C#'s useful functions is the Queue function. A queue of any type of object can be created. The program can then use the "Enqueue" or "Dequeue" methods to add or remove objects as it must. This is useful for keeping track of objects which need to be kept in order, which is a common occurrence in the conveyer system. Another useful function is the List function. Like the Queue, a list of any type of object can be created. This list can then be added to, searched, or removed from using inbuilt C# functions. The alternative to these two functions would have been to use arrays, however both the List and the Queue functions are easier to use, and can dynamically change size.

The Java programming language shares many similar functions to C#. It also has an advantage in that it is an open-source platform. Many discussion and help forums are also available on the internet to support Java. It is therefore difficult to say which platform is more suitable. One advantage that C# has over Java is that RS232 interfaces and drivers to interface with I/O devices are more readily available in C# than in Java.

A proper study would have to be done to determine which platform is best. However, as previously mentioned, it is part of the ongoing MADRG research to determine C#'s suitability to holonic programming.

5.4 Common Aspects

5.4.1 Basic Holonic Structure

A holonic program framework was developed by Professor Anton Basson. This framework has built-in features which, after some study, simplify the process to construct a holonic control program.

Each holon inherits a basic holon structure from a holon base class. Each holon executes the following functions in the order shown in Figure 23.

- Start Up
- Process Inbox
- Handle Agenda
- Execute
- Handle Low Level Interface
- Shut Down

The flow diagram for these functions is presented in Figure 23.

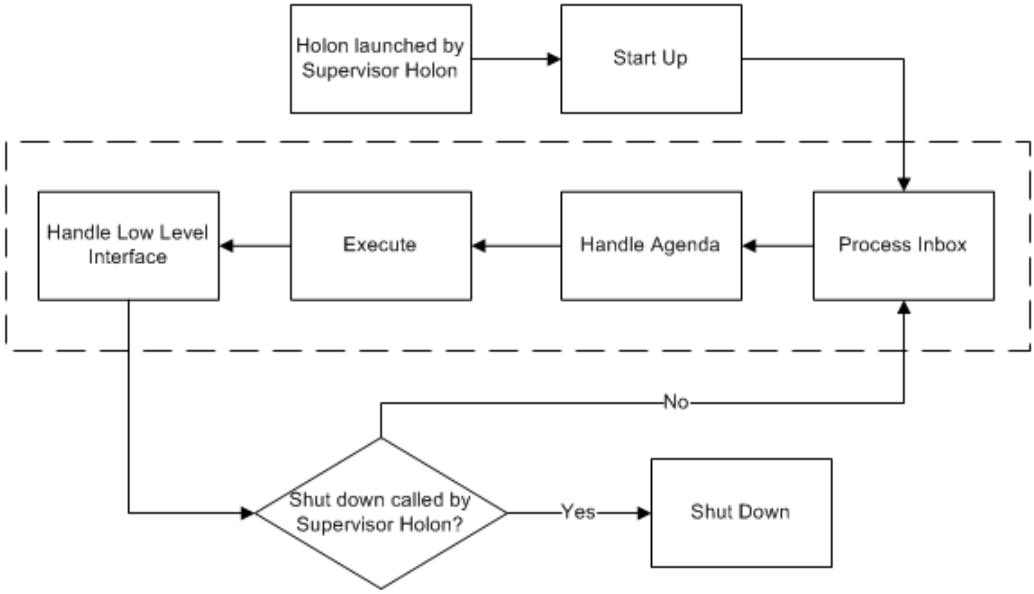


Figure 23: Basic holon structure flow diagram

The functions in the dashed box make up the main loop of the holon, these functions are continuously repeated unless the Shut Down is called by Conveyer Controller Holon. The functions are explained as follows:

- Start Up: This function is only executed once when the holon is launched. It consists of all operations which need to be performed before the holon can enter the main loop, such as initialisation of functions and members and initiating communications between low level interfaces.
- Process Inbox: All received messages stored in the inbox are processed. The inter-holonic messaging is explained further in Section 5.4.2.
- Handle Agenda: The holon assigns a ranking to each of the tasks it needs to perform and assembles them in an agenda.
- Execute: The holon executes the tasks it needs to perform, in the order the agenda prescribes.
- Handle Low Level Interface: The holon communicates with any low level resources it needs to communicate with.
- Shut Down: This function is executed when the holon is terminated. Communications are safely closed and all operations are stopped in a safe manner. The shut down function can only be called when all other functions are complete.

Note that the holons implemented for the conveyer control program do not use the Handle Agenda and Execute functions, and only the Operational Holons use the Handle Low Level Interface functions. This is because almost all operations are immediately performed on the receipt of a message, which is done by the Process

Inbox function. Operations also do not take so long that an agenda is required to prioritise them.

Each holon runs in its own thread. This is initiated in the base holon class. All holons must also add themselves to a Directory Facilitator on start up. This is used to keep track of the holons, and for holons to request services from other holons in the Directory Facilitator.

5.4.2 Inter-Holonic Communication

A crucial part of a holonic program is that the holons must be able to communicate with each other. A message class was developed to facilitate this. Each holon has an inbox (i.e. a message queue), and every message received by a holon is logged in its inbox, which can then be processed in the Process Inbox function which is part of the main loop. All messages sent between holons can also be displayed on a HMI to assist with debugging.

All message types are inherited from the parent message class. Type-specific payloads can then be defined, and holons can execute certain functions based on the type of message received as well as the payload of the message. A glossary of all types of messages is included in Appendix A.

5.5 Description of Holons

The different types of holons will be explained in this section. Note that the function and operation of a holon may only be clear when the description of the holons it interacts with is also read.

5.5.1 Conveyer Controller Holon

The ConC fulfils the supervisor requirement of ADACOR. On start-up, the ConC launches all the required holons of the control program. It also sends all start up data required by the other holons to the respective holons. For example, when a PH is launched, the ConC provides it with its starting location, its destination and the instance of the ConMap it needs to communicate with. On HLC programme shut down, the ConC is responsible for safely terminating all other holons and only shuts down once all the other holons have shut down. The ConC also has the ability, if required, to terminate holons while the program is still running. This is used when a PH has completed its task and been returned to the pallet magazine. The ConC receives tasks from either the cell controller or the HMI.

5.5.2 Conveyer Map Holon

The ConMap is responsible for calculating the route the pallets need to take and keeping track of all the pallets on the conveyer system. When the ConMap is launched, it receives the conveyer setup data it requires from the ConC. The ConMap only uses its Process Inbox function, therefore only acts when other holons communicate with it.

The ConMap keeps track of holons using a queuing system. Each OH sector is assigned a queue. A sector is the path between consecutive nodes of the conveyer. Each node will have a module associated with it, which in turn is controlled by an OH. Once a pallet has left an OH's sector, by passing through the module the OH controls, the OH sends a message to the ConMap informing the ConMap of this. The ConMap then removes the pallet from the front of the sector queue, and then adds that pallet to the back of the next sector queue. The pallets are passed on from sector to sector using this method. The ConMap then also sends a message to the next OH notifying it that a new pallet has entered its sector.

The ConMap is also responsible for calculating the route a pallet must take. The PH assigned to each pallet sends a message requesting a route, giving the ConMap a starting and end location. The ConMap then executes a path finding function, based on Dijkstra's algorithm, to determine the shortest route. It then also determines the OHs the PH needs to contact in order to complete its task. The ConMap sends a message back to the PH containing the route and OHs the PH must contact. These are in the form of data arrays which form part of the payload of the message the ConMap sends to the PH.

It was originally planned to separate the route finding and conveyer status functions into two separate holons. However, this would have resulted in two very basic holons and an unnecessary amount of inter-holon communication. As stated in the introduction, optimisation of the conveyer is not the objective of the project; however a future project is planned whose objective is to optimise the pallet movements. Consolidating path planning and conveyer status management into one holon should simplify the optimisation process.

5.5.3 Pallet Holon

The PH is a combination of the conventional task holon and operational holon. Every pallet that is launched and active on the conveyer has a PH assigned to it. It has some task holon characteristics in that it performs the task of moving a pallet from point to point. However, it is also similar to an operational holon because it is tied to a physical entity, that of the pallet.

When the PH is launched, it is given the current location of the pallet it is assigned to by the ConC. The ConC will also send a message instructing the PH to move the pallet to a specified location on the conveyer. The PH will then send a message to the ConMap requesting a route to the PH's intended destination, which the ConMap will reply with and the PH will then store. The PH is then responsible for guiding the pallet through the conveyer to its final destination. While a pallet is on the conveyer, there is always a PH assigned to it.

As the pallet leaves a sector and enters the next sector, the ConMap notifies the PH of this transition. The PH will then make a reservation with the OH whose sector it is now entered. This reservation will tell the OH to either let the pallet pass through on a default route, or perform some other action on it, depending on the OH type.

On completion of a PH's task, which is the guidance of a pallet the PH is assigned to from a start point to a destination point, the pallet will be at one of the destination type modules, that being either the lifting unit module or the lifting unit with transverse conveyer module. The PH will continuously check its inbox queue until it has received a new instruction form the ConC. If there is no task for the pallet, the operator can instruct the ConC to send a message to the PH instructing the PH to move the pallet to the pallet magazine, where it will be removed from the conveyer. Once the pallet has been removed from the conveyer, the ConC terminates the PH.

5.5.4 Operational Holon

The OH is responsible for the direct management of the physical resources of the conveyer system, which are the LLC operated modules. There are different types of OHs, depending on the type of module it is designed to operate. The different types are:

- Lifting Unit (destination type module)
- Lifting Unit with Transverse Conveyer (destination type module)
- Divert Unit
- Divert Unit with Pallet Magazine

The OH names are the same as that of the physical modules that they are designed to control, which were discussed in Section 4.4.

In the programming structure, there is an operational holon parent class, which includes all the functions that the different types of OHs share. The OHs for the specific modules then inherit the parent class as part of their structure.

On start-up, the OH and LLC connect to each via TCP/IP communication. The OH executes a server listening function on every main loop cycle, to hear if the LLC's status has changed. If there is a change in status, the OH executes functions based on the change in status.

Each OH has a queue which is uses to track pallets as they enter its sector. The OHs assigned to the divert unit with pallet magazine module and divert unit module have more than one path leading to them, therefore more sectors (and corresponding sector queues) to monitor. The OH adds a pallet to the queue on notification from the ConMap. It will then also remove a pallet from the queue if it leaves the OH's sector. The OH will then inform the ConMap that a pallet has left one of its sectors.

Each OH has a list of reservations. A reservation is a separate class which consists of the PH which made the reservation, and the actual task the OH needs to execute for the PH. All reservations are kept in one reservation list, regardless of what sector the pallet is expected to be in.

For the destination type OHs, the reservation request is either to pass the pallet through, or to lift the pallet. There are only two possible options, which is straightforward. For the divert type OHs, which have multiple input paths, it is more complicated. The PH needs to inform the OH which node it wants to be heading towards after it has left the OH's sector. The OH will then check if the requested node is a viable request using the Direction Number Array explained further in 5.7.3. If the request is valid, the OH makes the proper reservation and stores it in the reservation list. If the request is invalid, for example the PH requests a destination which the OH cannot find, an error message is displayed, and the pallet is passed through on the default route so that the other pallets are not blocked,

When a pallet trips the module's proximity switch, the OH checks what pallet is at the end of the queue to determine its identity. It then searches the instruction list to find the reservation the PH has made for that pallet. If the PH has not made a reservation, then there is an error condition, and the HMI will display an error message. The OH will execute a default function, which will be letting the pallet pass through on a default route.

Similarly, if a module detects the proximity of a pallet, when there should not be a pallet in the OH's sector, then there is an error condition. The OH will let the pallet pass through on the default path so that the other pallets are not blocked. The OH will send a message to the ConMap that an unknown pallet has passed through. An error message is sent to the ConMap and the HMI, which notifies the operator.

Note that the ConMap holon keeps track of all the sectors on the conveyer, using queues. Each OH also keeps track of its own sectors using queues. The pallet location information is therefore duplicated in the ConMap holon and in all the OHs.

This potentially problematic, because a likely error is that the pallet location information in an OH does not match that held in ConMap. There are also redundant messages containing the same information being sent to OHs and ConMap. At the time of program implementation, duplicating the information seemed the most straightforward option. Any difference between the ConMap pallet information and OH pallet information would then signal an error. Therefore, it could be used as a diagnostic tool. However, in hindsight, it would have been better to have a one central pallet location database, which all holons can periodically read but only the ConMap can write to.

5.6 Human Machine Interface

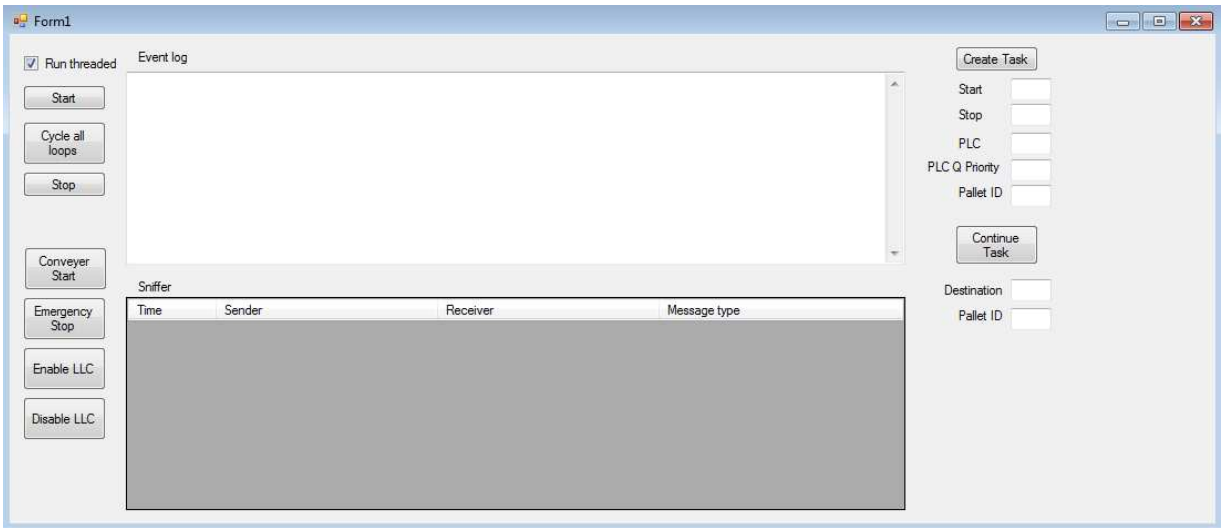


Figure 24: HMI command window

Figure 24shows the HMI for the HLC. Note that for testing and demonstration purposes, the HMI replaces the Cell Controller as the source of instructions for the conveyer control system. If implemented as part of a manufacturing cell, the conveyer control system would receive instructions directly from the Cell Controller. During testing, the Cell Controller was not yet ready to be implemented as part of the entire setup.

The Event Log box can be used by any holon to inform the user of a significant event. It is particularly useful for debugging. The Sniffer box records and displays all messages sent between holons, which is also a critical debugging tool.

The Start button starts the conveyer controller by launching the ConC, which subsequently launches all the other holons. The Stop button calls the shut down procedure executed by the ConC explained in Section 5.5.1. The Run Threaded check box and the Cycle All Loops button are debugging tools. If the Run Threaded box is not checked, the holons only execute one circuit of their main loop cycles every time the Cycle All Loops button is pressed.

The Conveyer Start button is used to tell all LLCs to go into Stop and Check mode. It also starts the main conveyer track moving. The Emergency Stop tells the PLCs to immediately stop all actions, by telling them to go into the Passive state. The Disable LLC button tells the OHs to stop communication with the LLCs. The Enable LLC button does the opposite.

On the right side of the window, the Create Task button is used during development to launch a new pallet and Pallet Holon. However, if the information boxes are not correctly filled in, an error message is displayed. The required information is the start node, the destination node, the PLC in whose sector the pallet is in, the number of the sector according to the OH assigned to that PLC

(referred to as the PLC Q Priority), and a user defined pallet ID number. The reasons for this information are explained in Section 5.7.

The Continue Task button is used to tell a PH to move a pallet to a new destination. The required information is the Pallet Holon ID number and the new node destination.

5.7 Conveyer Data Structures

The programming for the holons described in Section 5.5 is independent from the setup of the conveyer, with the possible exception of the Operational Holons. If the physical makeup of the module the OH is assigned to changes, the OH programming in the HLC may have to be changed, or an entirely new OH will be programmed if the change is drastic enough.

To describe the data structures, consider the conveyer shown schematically in Figure 25, which is a schematic representation of Figure 8. Figure 25 shows the node identification numbers and PLC identification numbers in oval text boxes. The square call-out labels contain the distances between each node. The distance measurement unit is approximately how many pallets would fit in the sector between nodes.

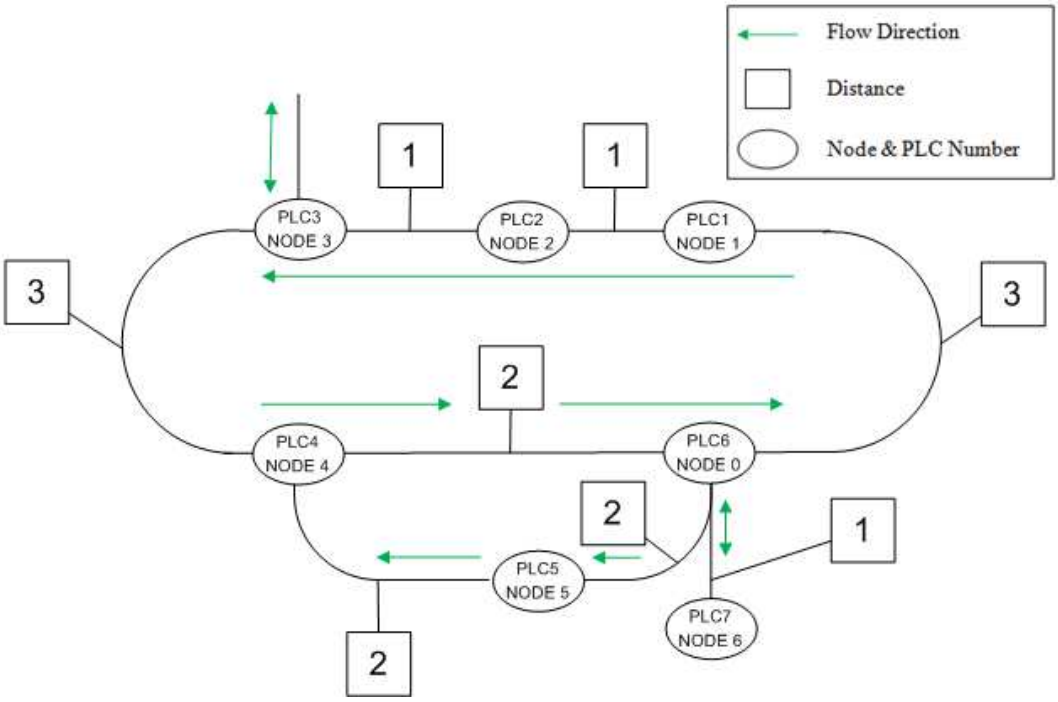


Figure 25: Conveyer setup diagram

Note that PLC 6 and PLC 7 and Node 0 and Node 6 are both part of the divert unit with pallet magazine module. It was initially thought that the pallet magazine and the module would be operated by the same PLC. However, the pallet magazine

already had a Siemens PLC installed to operate it. It therefore made more sense to split the control up. As a result, a node was also created for the pallet magazine.

Ideally, the PLC identification numbers and the node identification numbers should have matched, as this would have simplified programming implementation. The PLC identification numbers were physically inscribed on the units and therefore could not be changed. In hindsight, the node identification numbers could and should have been changed to match the PLC numbers once the conveyer setup and conveyer sections had been finalised.

The content of the following data structures depend on the setup of the conveyer, and in the following section, the values are for the setup shown in Figure 25. If the setup of the conveyer changes, these are the data structures an operator would have to change for the HLC programme to work.

5.7.1 Conveyer Map Matrix

This matrix (illustrated in Table 7) contains the length of a sector between the nodes in the conveyer. It is approximately measured in how many pallets would fit in the sector. It is essential for computing the shortest distance between two points, which is done by the Dijkstra's algorithm. The rows of the matrix correspond to the start nodes, and the columns to the destination nodes. Because most of the conveyer is unidirectional, the matrix is not symmetrical. The only bidirectional path is that between nodes six and zero, and between nodes zero and six. Therefore, elements 0:6 and 6:0 are equal in value.

The elements which have no number assigned signify that there is no direct path between those two points. The HLC programme assigns an extremely high number in those spaces, so that the path finding algorithm only finds legitimate routes.

Table 7: Conveyer node mapping

	Destination Node Number							
Start Node Number		**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0		3				2	1
	1			1				
	2				1			
	3					3		
	4	2						
	5					2		
	6	1					1	

5.7.2 Conveyer LLC Matrix

This matrix (illustrated in Table 8) is similar to the Conveyer Map Matrix, but, instead of the elements containing distances between nodes, it contains the ID number of the OH which is responsible for that sector between nodes. It is used by the PH to determine which OH needs to be contacted to request a service from the OH.

Table 8: Conveyer LLC mapping

	Destination Node Number							
Start Node Number		**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0		1				5	7
	1			2				
	2				3			
	3					4		
	4	6						
	5					4		
	6	6					5	

5.7.3 Direction Number Array

This array is only used by the non-destination type OHs, whose task it is to divert the pallets. On receipt of a task request from a PH, the OH will scan through the array to determine if the requested destination of the PH matches the number contained in one of the elements. If it does match, the OH will use the associated column number to determine what operation needs to be performed.

In Table 9, the second row shows the possible node requests for the divert unit with pallet magazine module, stationed at Node 0 and with PLC number 6 in

Figure 25.When a PH makes a task request, it would say, for example, that its next destination node is node 6. The OH assigned to the divert unit with pallet magazine module will then search the row of the array for the number 6. When it finds the number 6, it will look at the corresponding column number, denoted in Table 9 as an instruction number, which is the number 2. Using the instruction number and the number of the sector the pallet is arriving on, the OH knows which instruction to give to the LLC in order to send the pallet to its intended destination.

In the schematic shown in Figure 25, only the divert unit with pallet magazine module has multiple output paths, therefore it is the only OH which requires this array. However, in the second setup shown in Appendix B in Figure 29, the divert unit module also has multiple outputs. It therefore, also requires an array such as that shown in Table 9.

Table 9: Direction number array

	Instruction Number		
	0	**1**	**2**
DU&PM	1	5	6
	Possible Destination Node Request		

5.7.4 Incoming Queue Array

At start up, all the queues for each sector are instantiated by the ConC. They are then assigned to the Incoming Queue Array and the Outgoing Queue Array (Section 5.7.5). On ConMap launch, these arrays are passed to the ConMap. When an OH sends a message to the ConMap informing the ConMap that a pallet has left its sector, the ConMap will use the Incoming Queue Array to remove the PH from the correct sector queue, based on the OH ID number and the number of the sector according to the OH, which is part of the message payload.

The divert unit module and divert unit with pallet magazine module both have more than one entry point. They therefore have more than one sector to monitor. The sectors are numbered, and when the divert unit module or divert unit with pallet magazine module informs the ConMap of a pallet departure, a sector number is also required, so that the ConMap knows which one of the sector queues it must remove a PH from.

The naming convention for the sector queues is the departure node number, underscore, and then the destination node number. The row numbers are the OH ID numbers, and the columns are the sector numbers according to the corresponding OH.

Table 10: Incoming queue array

		Incoming Sector Number	
		0	**1**
Operational Holon ID	**1**	Zero_One	
	2	One_Two	
	3	Two_Three	
	4	Three_Four	Five_Four
	5	Zero_Five	
	6	Four_Zero	Six_Zero
	7	Zero_Six	

5.7.5 Outgoing Queue Array

The Outgoing Queue Array (Table 11) is similar to the array described in Section 5.7.4. The difference is that the ConMap will use this array to add a PH to the correct queue. The OH transition message contains a sector number for the output path that pallet is sent on, which the ConMap will use to identify the correct queue.

In this conveyer setup, only the divert unit with pallet magazine module has more than one exit, however, in the second setup, the modified divert unit module also has multiple exit points. Refer to Appendix B for the second conveyer setup.

Table 11: Outgoing queue array

	Outgoing Sector Number			
Operational Holon ID		**0**	**1**	**2**
	1	One_Two		
	2	Two_Three		
	3	Three_Four		
	4	Four_Zero		
	5	Five_Zero		
	6	Zero_One	Zero_Five	Zero_Six
	7	Six_Zero		

5.7.6 Outgoing PLC Connect Array

The array shown in Table 12 is used as a directory by the ConMap to contact the next OH which is due to receive a pallet. Part of the payload of the transition notification message from the OH to the ConMap is the ID number of the sending OH, and the sector number of the path the pallet is being sent on. The ConMap

then sends a pallet proximity notification message to the next OH by using the sender ID number to go to the correct row, and using the sector number to go to the correct column. The corresponding element contains the ID number of the OH the ConMap needs to send the pallet proximity message to.

Table 12: Outgoing connect array

	Outgoing Sector Number			
Operational Holon ID		**0**	**1**	**2**
	1	2		
	2	3		
	3	4		
	4	6		
	5	4		
	6	1	5	7
	7	6		

5.7.7 Outgoing Sector Number Array

The array shown in Table 13 is similar to that described in 5.7.6. The difference is that it contains the number of the sector receiving the pallet, according to the receiving OH.

This is necessary because in the case where some OHs have multiple input paths, and therefore more than one sector to monitor, the ConMap needs the data in this array to inform the OH which sector queue it needs to add a PH to.

For example, OH 5 sends a transition message to the ConMap with Outgoing Sector Number 0. The ConMap then sends the contents of the array element corresponding to OH 5 and rank number 0 to the next OH, which is the number 1. The next OH then knows that it is receiving a pallet on its number 1 input sector.

Table 13: Outgoing sector number array

	Outgoing Sector Number			
Operational Holon ID		**0**	**1**	**2**
	1	0		
	2	0		
	3	0		
	4	0		
	5	1		
	6	0	0	0
	7	1		

5.8 HLC Data Structure and Communication Example

This section shows an example of how the data structures discussed in Section 5.7 are used, based on the conveyer schematic shown in Figure 25.

A PH is leaving Node 5, and wants its final destination to be Node 6. The following steps will be taken:

1. OH 4 sends a Pallet Departure message to the ConMap. The payload is the OH's ID number (4), the departing pallet, the Incoming Sector Number (1), and the Outgoing Sector number (0).
2. The ConMap goes to the Incoming Queue Array (Table 10) and removes the pallet from the front of the Queue corresponding to OH ID 4 and Incoming Sector Number 1, which is queue Five_Four.
3. The ConMap goes to Outgoing Queue Array (Table 11) and adds the departing pallet to the queue corresponding to OH ID 4 and Outgoing Sector Number 0, which is queue Four_Zero.
4. The ConMap sends a Pallet Proximity Update to OH 6. It knows it must send a the message to OH 6 by going to the Outgoing Connect Array (Table 12), going to the row which corresponds to OH 4 and the column which corresponds to Outgoing Sector Number 0, which gives the OH ID number of 6. Part of the message payload is the sector of OH 6 which the pallet is entering. To determine this, the ConMap uses the Outgoing Sector Number Array (Table 13). Using the same procedure as that used with the Outgoing Connect Array, the number will be 0.
5. OH 6 receives the Pallet Proximity Message from the ConMap. It uses the data from the payload to add the pallet to the queue which corresponds to its 0 sector.

6. The ConMap sends a Transition Notification message to the PH assigned to the pallet.
7. The PH sends a Task Request Message to OH 6. The payload is the destination where the pallet wants to be after it has left Node 0, which is Node 6.
8. OH 6 receives the Task Request Message. It uses the Direction Number Array (Table 9) to determine what operation number it needs to perform when the pallet reaches the OH, which will be operation number 2. The OH stores the operation number and the PH ID number together in a list.
9. When the pallet trips the proximity switch of OH 6, OH 6 will compare the ID number of the PH at the front of that sector queue to the list of PHs it has. When OH 6 has found a match, OH 6 will execute the required operation, and the process will start again.

6 IMPLEMENTATION AND EVALUATION

6.1 Implementation

Prior to formal testing, each of the modules was individually tested to determine whether they were functional. A test program was written which allowed for manual connection and communication between the PLCs and the user. Modifications to the PLC programs were made as required. Modules were added one by one to the conveyer system, before the entire system was tested as a whole.

It was noted that it takes approximately on average four seconds for each PLC to connect to the HLC via the TCP/IP connection. Sometimes, the PLC would fail to connect, and the system would have to be restarted before it would connect again. This is due to built in timeouts in the PLCs. If a no connection took place within a certain amount of time, the PLCs would stop trying to connect and stop trying to receive connections.

6.2 Current Holonic Implementation

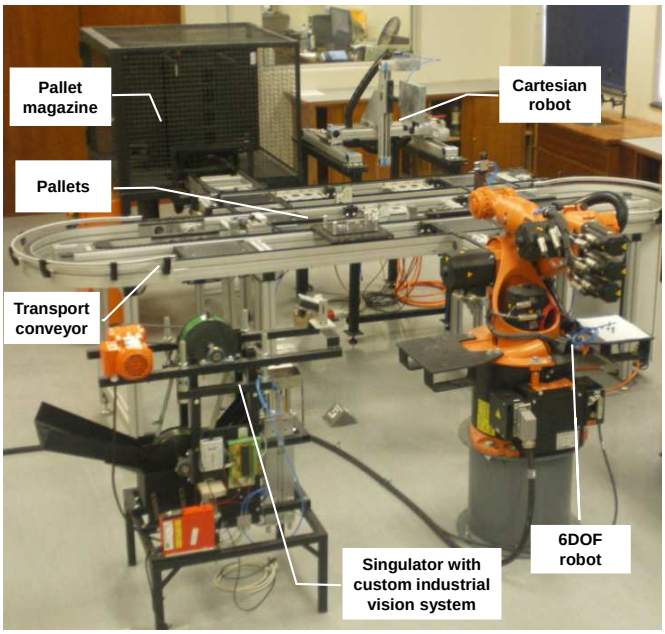


Figure 26: Holonic laboratory setup

A previous setup in the Automation Laboratory is shown in Figure 26. In this example, the Cartesian robot, 6 DOF robot, singulator and transport conveyer would be considered as either resource or operational holons, depending on whether the PROSA or ADACOR architecture is used. The entire process would be controlled by a Cell Controller, which can be regarded as a supervisor holon.

Note that each subsystem, or resource or operational holon, could in itself have an entire holonic structure complete with the various types of holons which make up a holonic architecture. This is similar to the fractal nature explained in Figure 4.

6.3 Convertibility Experiment

Tests were conducted to determine the reconfigurability of the system. The only comparable data available was that of Le Roux (2013) who previously worked on the conveyer system. The laboratory technician, Reynaldo Rodriguez, who had assisted Le Roux in the conveyer setup of his project, was questioned and gave his opinion as to whether the new system was an improvement on reconfigurability or not. In short, Rodriguez felt that the modular, distributed control approach has improved the system's reconfigurability when compared to previous work. Rodriguez did, however, say that there are additional start-up costs required for this approach when compared to a centralised approach. His full opinion is recorded in Appendix B.

The major reconfigurability test was the physical hardware change to the conveyer. A new conveyer setup was determined, and Rodriguez and the author did the physical change.

Figure 27 shows the original conveyer setup, as well as pallet flow directions. The second setup is shown in Figure 28. The modifications are:

- A lifting unit module was moved from the parallel conveyer to the main conveyer. The lifting unit module is now between the divert unit and the divert unit with pallet magazine modules.
- The lifting unit with transverse conveyer module was moved from the main track to the parallel track, replacing the lifting unit module which has been moved.
- The parallel track conveyer direction has changed.

These modifications required modifying the divert unit module and divert unit with pallet magazine module to handle the changed conveyer direction and change in proximity sensors. The PLC program for the LLC and the C# program for the HLC for both the divert unit and divert unit with pallet magazine modules were modified to handle the change, because the physical structure of both modules had changed.

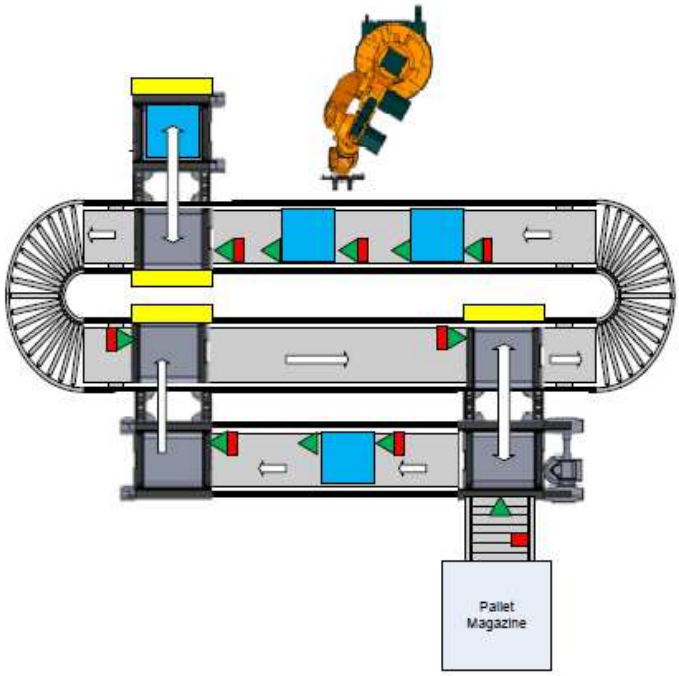


Figure 27: Conveyer setup 1

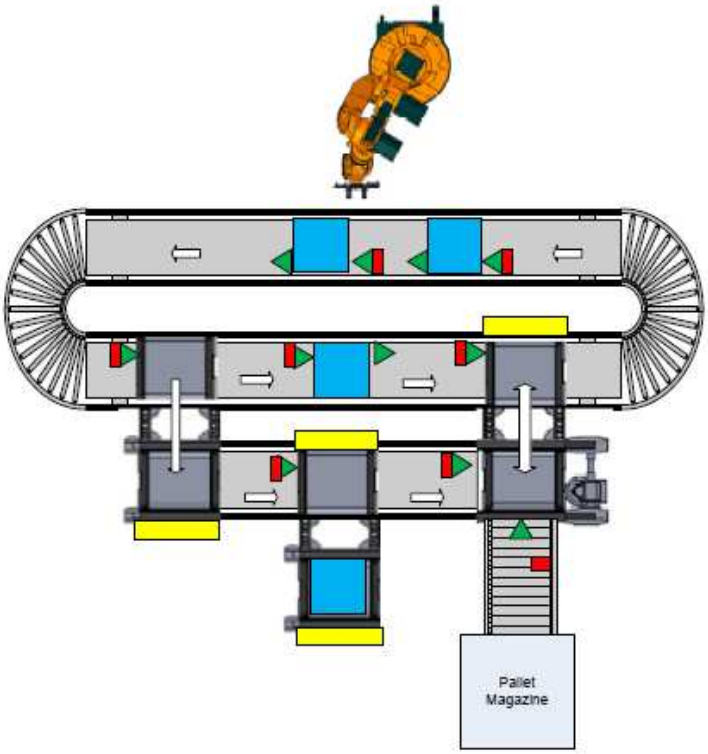


Figure 28: Conveyer setup 2

7 CONVERTIBILITYEXPERIMENT DATA ANALYSIS

The physical hardware reconfiguration took a total of 8 man hours. The work was done by Rodriguez, who is considered an experienced and skilled lab technician, and the author, who assisted under Rodriguez's supervision where required. In total, the operation took 4 hours to complete.

According to Rodriguez, this is a significant improvement on the previous physical reconfiguration, which took approximately two weeks to change. This proves that the distributed LLC control system is an improvement on centralised LLC control.

The software adjustments to the LLC program and the HLC program to account for the module modifications took another 2 hours. The time taken to determine the new data matrices for the HLC took 1 hour. Trouble shooting the new software setup took 2 hours, as there was an error in the structure of the program which had not been discovered in the previous setup. Had there not been a structural error, trouble shooting should have taken half an hour.

In total, it took 5 hours to modify the software of the program. With more reconfiguration practice and without the structural software error, this time could be brought down considerably to approximately 3 hours. In total, 13 man hours were required for the entire reconfiguration operation.

7.1 Reconfiguration Characteristic Conformity

This section discusses to what extent the new design conforms to the six characteristics of reconfigurability, as discussed in Section 2.1.

Modularity: Because of the distributed control, the design can be considered modular. As previously explained, the conveyer was separated into modules, each with a controlling LLC and an operational holon assigned to each module.

Integratibilty: A further test on integratibilty would have been to have different types of LLCs operating the modules on the conveyer. The one different LLC is the Siemens PLC controlling the pallet magazine, and communication between the pallet magazine and the HLC was not achieved. However, the HLC programme is written in C#, which is a ubiquitous programming language. The communication is also done using TCP/IP, which is also a commonly used communication protocol. These two factors enhance the system's integratibilty.

Customization: The customization test was the modifying of the divert unit module and the divert unit with pallet magazine module. There is no data to compare the test to, which makes it difficult to assess this characteristic. However, the modification was carried out with little difficulty, according to Rodriguez. The software design which makes use of inheritance and a standard holon structure also aids software modification, because a user does not need to design a

completely new holon, but can rather build on the parent classes and standard holon structure.

Convertibility: The system demonstrated its convertibility during the reconfiguration testing. The data available to compare with is that of Le Roux (2013), and according to Rodriguez it was substantially easier to do the physical hardware change than what it was previously. While it is difficult to judge whether the system is truly convertible, this system at least offers an improvement on the previous system.

Diagnosability: This is one characteristic where the design is lacking. The only diagnosable functions are messages sent to the HMI to say that a pallet is either not supposed to be in a certain sector, or that a PH has failed to make a booking. Possible improvements are discussed in Section 8.

Scalability: Because of the system's modular design, scalability is achieved. During implementation of the system, modules were brought online one by one. A test was also performed where a configuration was used which excluded certain modules installed on the conveyer. It was tested to determine functionality, and immediately after testing, a new configuration was loaded using all modules, and tested again. The system was functional during both tests, demonstrating scalability.

7.2 Pallet Management Results and Analysis

Testing was done to determine whether the system can successfully navigate multiple pallets from point to point. These tests were conducted on both setups, and during scalability testing. The procedure was to start small, with just a single pallet on the system, and then to add more and more pallets until the system was flooded with pallets. Modules with multiple entry and exit points were also tested with multiple pallets to determine if the system could successfully handle more than one pallet.

Both setups were able to successfully manage multiple pallets. Pallets were launched from the pallet magazine and placed at various points on the conveyer and given instructions to move to a location. However, when the system was flooded, the system was not aware that there were too many pallets in a certain sector, leading to an overflow of pallets, which resulted in errors. This could be rectified by putting a limit on the number of pallets allowed into a sector, and an OH would then not be allowed to pass a pallet into a sector which is already full.

Pallets were also sent to the pallet magazine to determine if the pallet magazine could successfully remove pallets from the system. The pallet magazine was manually operated, but it was able to successfully remove pallets from the conveyer. The HLC program also successfully terminated PHs which were no longer operating on the conveyer.

Various scenarios were tested, and in some cases modifications were necessary. One such scenario was what would happen to the lifting unit with transverse conveyer module if the pallet it was holding was given a new destination at the same instance when the module is letting another pallet pass through. This would result in the module being sent a new instruction while in the process if executing another instruction. Initially, this caused PLC to freeze and no longer respond, possibly due to conflicting states and messages. The OH code was modified to not allow a state change instruction while the PLC is executing a pass pallet sequence, after which the problem no longer arose.

Initially, the lifting unit module did not have a sequence for releasing pallets. The original method to release pallets was to return to the Stop and Check state. However, if there was a pallet waiting immediately behind the module which had made a reservation to be lifted, the lifting sequence would be actuated before the first pallet had fully left the module, resulting in the wrong pallet being lifted. To resolve the problem, a release sequence was added to the control, and the module would only return to the Stop and Check state once the first pallet had successfully left the module.

There were also problems with the two divert unit with pallet magazine modules. During some of the divert sequences, pallets would sometimes be partly dislodged from the conveyer track, and would block the path as a result. The divert sequences were modified by adding timers which would allow the pallets to settle in position first before the next operation was carried out which improved the situation. However, on rare occasions, pallets are still sometimes dislodged. This problem could be rectified by slowing down the conveyer movement speed.

7.3 HLC Platform

The author has previous experience in C programming language and Java programming language, and once the initial stages of learning how to use the C# was complete, found C# easier to use than C or Java. However, this could just be because the author has more experience in learning new programming languages. The-built in functions discussed in Section 5.3 also assisted greatly with the HLC programming. While it is not possible to say which programming language is best suited to the project without testing them all, C# was more than adequate and could do all that was required from it.

7.4 Communication between HLC and LLC with TCP/IP

The TCP/IP communication performed satisfactorily. The only problem was the relatively long time it takes for the HLC to connect to the LLC, sometimes as long as approximately 10 seconds to connect in extreme cases. Also, it would sometimes not connect at all, even before the PLC’s connection timeout, and the LLC and HLC would have to be restarted. A possible cause of this is interference on the network due to other unrelated communication taking place.

Once the system was connected, the communications were stable and fast enough for conveyer control purposes.

7.5 Use of PLC as LLC

It is difficult to state whether the LSIS PLC is best suited to the task without comparing it to other PLCs. However, the LSIS PLC was reliable and straightforward to use, therefore adequate to the task required.

The LSIS PLC has many additional capabilities, such as PID control, temperature sensing and more advanced communication capabilities, amongst other features. It is therefore much more capable than what was required for the project, which only required I/Os, timers, and basic TCP/IP communication. An alternative, such as a microcontroller system, which has these capabilities, could be considered and tested as a more cost effective option.

8 IMPROVEMENTS AND RECOMMENDATIONS

As mentioned in Section 7.1, the diagnosability of the system can be improved. Self-diagnostic tests can be performed on start-up to determine whether the physical setup matches the software configuration. Such a test could be OHs sending a test message to neighbouring OHs, and checking that the correct OH responds.

Currently, the RFID read/write heads (mentioned in Section 1.3) are not used. Pallet management is done only using queues and dead reckoning. The RFID system could be used as a backup check to ensure that the pallets are where they are supposed to be. If there was a scenario where pallets changed order within the same sector, the system would not know the difference between the two pallets. The use of RFID read/write heads would eliminate this possibility. This would add to the diagnosability of the system.

An advanced way to determine the conveyer setup would be to make use of RFID readers. On system start-up, pallets could be sent on all possible paths on the conveyer. Using the RFID readers and the module proximity switches, a model of the conveyer could be built up and stored. If the RFID card readers are not used, the procedure could be still performed, however only one pallet would be used.

In the event that a module fails or a path becomes blocked, the system does not have a means to respond to that. An improvement would be to be able to dynamically change the conveyer map array to account for paths which have been blocked and therefore for practical purposes no longer exist. Dysfunctional holons and PLCs should also be able to be automatically restarted or worked on without disrupting the rest of the system.

In the case where a pallet is not accounted for in a sector or a PH fails to make a booking, the only error procedure is to notify the user. Ideally, a function should be created which automatically removes such pallets from the conveyer, without disrupting any of the other operations.

As previously mentioned, the conveyer sometimes becomes blocked due to a pallet being dislodged from the track, or a PLC crashes. Timeouts should be introduced by either the OH, the PH or both, which can then send warning messages if it takes too long for a pallet to reach its destination, or if an operation takes too long to complete.

The software setup makes use of many arrays. It may be possible that many of these arrays could be combined into multidimensional arrays, or left out altogether. Currently, a new software setup needs to be hardcoded into the HLC code. While the hard coding all takes place in one area of code, which simplifies modification, ideally, provision should be made for a data file to be created by a user, which can then be uploaded to the HLC programme on start-up.

A current significant factor in the South African industrial environment is power outages due to load shedding. If an outage occurred and the system was unexpectedly shut down, a robust system would be able to carry on from where it left off, with minimal diagnostics. This could be achieved by periodically saving the conveyer status to permanent memory, or if the RFID heads were in operation, perform a diagnostic circuit to find out where all the pallets are located, and then to carry on with the various tasks once all the pallets have been located.

9 CONCLUSION

The primary objectives of the thesis stated in Section 1.2 are the design and implementation of the control system for a reconfigurable multi-pallet conveyer system. The main feature of the system is that it had to be reconfigurable, and therefore had to conform to the characteristics which define reconfigurability. Optimisation of pallet movement was not an objective.

Study of previous research revealed that holonic architectures aid reconfigurability. A modified ADACOR holonic structure was chosen over the PROSA architecture, because there was a need for some central control in the form of a supervisor holon. Research was also done in various conveyer traffic control methods, which led to the use of Dijkstra's algorithm for path planning purposes. C# was shown to be suitable to the task of holonic software programming. Various inbuilt functions were particularly useful.

A decentralised control structure was used. Research was done into suitable LLCs. However, the choices were limited due to requirements for the case study. Multiple LLCs were used instead of one central LLC, which made the system modular, and therefore more reconfigurable.

Testing proved largely successful multi-pallet operation. Tests were also done to determine reconfigurability, such as the convertibility test, which proved that the distributed control paradigm was an improvement on centralised control. As mentioned in Section 7.1, the only reconfigurability characteristic which is not fully fulfilled is that of diagnosability. Various improvements to enhance diagnosability, amongst other things, are given in Section 8.

An important question is whether the costs of reconfigurability justify the usefulness of reconfigurability. To answer this question, the costs of this system would need to be compared to the cost of a centralised control system. It would also depend on how often the system would need to be reconfigured.

As mentioned in Section 7, the reconfiguration took a total of 13 man hours. If a small team of experienced personnel (three or four) work on the reconfiguration, a major reconfiguration should then be achievable in a one work day. A reconfiguration time of one work day could be suitable for a one week production run, where the system is reconfigured on day one, and then operated in that configuration for the rest of the week. Ultimately it is up to the user whether the additional costs justify the reconfiguration capabilities.

Further work would be the optimisation of the transport system. If resources allowed it, the system should be tested on much bigger conveyer system, or a simulation should be used.

REFERENCES

Black, G, and V Vyatkin. 2010. “Intelligent Component-Based Automation of Baggage Handling Systems With IEC 61499.” *IEEE Transactions on Automation Science and Engineering* 7 (2): 337–51.

Cormen, T, C Leiserson, R Rivest, and C Stein. 2009. *Introduction to Algorithms*. 3rd ed. Cambridge: The MIT Press.

ElMaraghy, H. 2006. “Flexible and Reconfigurable Manufacturing Systems Paradigms.” *International Journal of Flexible Manufacturing Systems* 17 (4): 261–76.

Koren, Y, U Heisel, F Jovane, T Moriwake, G Pritschow, G Ulsoy, and H Van Brussel. 1999. “Reconfigurable Manufacturing Systems.” *Keynote Papers* 48: 527–40.

Koren, Y, and M Shpitalni. 2010. “Design of Reconfigurable Manufacturing Systems.” *Journal of Manufacturing Systems* 29 (4). Elsevier Ltd: 130–41.

Le Roux, A. 2013. “Control of a Conveyor System for a Reconfigurable Manufacturing Cell.” MScEng (Mechanical) thesis, Stellenbosch University.

Leitão, P, and F Restivo. 2006. “ADACOR: A Holonic Architecture for Agile and Adaptive Manufacturing Control.” *Computers in Industry* 57 (2): 121–30.

Mehrabi, MG, AG Ulsoy, and Y Koren. 2000. “Reconfigurable Manufacturing Systems: Key to Future Manufacturing.” *Journal of Intelligent Manufacturing* 11: 403–19.

Meng, F, D Tan, and Y Wang. 2006. “Development of Agent for Reconfigurable Assembly System with JADE.” *Proceedings of the 6th World Congress on Intelligent Control and Automation*, 7915–19.

Michaelis, M, and E Lippert. 2013. *Essential C#*. 5th ed. Upper Saddle River, New Jersey: Addison-Wesley.

Paolucci, M, and R Sacile. 2005. *Agent-Based Manufacturing and Control Systems*. London: CRC Press.

Van Belle, J. 2013. “A Holonic Logistics Execution System for Cross-Docking.” PhD Eng (Mechanical) thesis, KU Leuven.

Van Belle, J, B Saint Germain, P Verstraete, P Valckenaers, O Ali, H Van Brussel, and D Cattrysse. 2009. “A Holonic Chain Conveyor Control System : An Application.” *HoloMAS 2009*, 234–43.

Van Brussel, H, J Wyns, P Valckenaers, L Bongaerts, and P Peeters. 1998. "Reference Architecture for Holonic Manufacturing Systems: PROSA." *Computers in Industry* 37 (3): 255–74.

Versteegt, C, and A Verbraeck. 2002. "Holonic Control of Large-Scale Automated Logistic Systems." *IEEE 5th International Conference on Intelligent Transportation Systems*, 898–903.

Vrba, P, and V Marík. 2010. "Capabilities of Dynamic Reconfiguration of Multiagent-Based Industrial Control Systems." *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 40 (2): 213–23.

Weber, A. 2008. "AGVs vs. Conveyors." *Assembly Magazine*[Online]. Available: http://www.assemblymag.com/articles/85767-agvs-vs-conveyors [28th July 2014].

APPENDIX A: MESSAGE TYPES

The following is a list of all message types. The naming convention is first to have a B, to distinguish the message type from messages which are part of the original holon program, followed by "Message" to show that it is a message. Next is the sender holon abbreviation, followed by the receiver holon abbreviation, followed by a short description of the message. Note that CMH is the Conveyer Map Holon. Also note that "Supervisor Holon" is a synonym for "Conveyer Controller Holon" and is indicated by "SH" in the message names.

A1: Messages sent by Conveyer Map Holon

BMessageCMH_OH_ProximityUpdate:

Message from the CMH to an Operational Holon. Its objective is to inform the Operational Holon, which in turn is responsible for the operation of a PLC that a pallet is in the PLC's area of interest. The Operational Holon will then assign the pallet to a queue which holds the order of pallets in the PLC's area of interest. The payload is the sector number and the instance of the PH assigned to pallet in proximity.

BMessageCMH_PH_CalculatedRoute

Message from CMH to a Pallet Holon. The payload is the array of nodes a pallet must transit to reach its destination, and an ordered array of Operational Holons the Pallet Holon must contact on its route.

BMessageCMH_PH_TaskComplete

CMH informs PH that it has reached its final destination.

BMessageCMH_PH_TransistionNotification

CMH informs PH that the PH has moved to a new sector.

A2: Messages sent by HMI

BMessageHMI_SH_ContinueTask

Message from HMI telling the SH that a specific PH needs to move from its current location to a new location. Payload is the PH's ID number and the destination node number.

BMessageHMI_SH_ConveyerEmergencyStop

Message from HMI to SH telling the SH to send a message to all OHs telling them to immediately stop all operations. All OHs will then change from whichever state they were in to the “Passive” state.

BMessageHMI_SH_ConveyerStart

Message from HMI to SH telling the SH to send a message to all OHs telling them to start the conveyer. All OHs then change from the “Passive” state to the “Stop and Check” state.

BMessageHMI_SH_DisableLLC

Message from HMI to SH telling the SH to send a message to all OH’s telling them to disable communication with low level controllers, that being the PLCs.

BMessageHMI_SH_EnableLLC

Message from HMI to SH telling the SH to send a message to all OH’s telling them to enable communication with low level controllers, that being the PLCs.

BMessageHMI_SH_NewTaskUserDefinedLocation

Message from HMI to SH informing the SH that a new pallet is on the conveyer. The SH is required to launch a new PH for the new pallet. The payload is the start point, the end point, the starting OH, the rank of the OH’s queue it is in, and the new pallet’s ID number.

A3: Messages sent by Operational Holons

BMessageOH_CMH_PalletDeparture

Operational Holon informs CMH that a pallet has departed its area of interest. The payload is the departing pallet, the PLC ID number, the sector number the pallet was received on, and the sector number the pallet is being sent on.

BMessageOH_CMH_PalletHasReachedDestination

OH informs CMH that a pallet has reached its destination. The payload is the OH ID number and the instance of PH which has reached its destination. On destination type modules will use this type of message.

BMessageOH_CMH_ReleasingPallet

OH informs CMH that it is releasing a pallet. Only an OH such as a Lifting Unit or Divert with Lifting Unit can send this message, because they are destination type modules. The payload is the OH ID and the instance of the PH assigned to the pallet being released.

BMessageOH_CMH_UnregisteredPalletDeparture

OH informs CMH that an unregistered pallet has arrived and left the OH.

BMessageOH_CMH_UnregisteredTaskDeparture

OH informs CMH that a pallet which did not have a corresponding task registered to it has arrived and left the OH.

BMessageOH_SH_PalletInMagazine

OH informs SH that a pallet is in the pallet magazine. Only the pallet magazine OH will send this type of message. On receipt of this message, the SH will shut down the PH associated with that pallet. The payload is the PH instance associated with the pallet in question.

A4: Messages sent by Pallet Holons

BMessagePH_CMH_RouteRequest

PH requests a route from the CMH. Payload is the start node and destination node.

BMessagePH_OH_ReleaseRequest

PH requests the OH which is holding it to release it.

BMessagePH_OH_TaskRequest

PH makes a booking at an OH for what the OH must do when the PH's pallet arrives at the OH. The payload is if the OH is the destination or not, the node where the pallet has come from, and what the next node the PH wants to go to is.

A5: Messages sent by Supervisor Holon

BMessagePH_SH_TaskComplete

PH informs SH that it is complete with the task it was assigned. In other words, it has reached the destination it was supposed to get to.

BMessageSH_CMH_StartUpData

SH sends the CMH all the data it needs to perform its operations. The data forms the payload for the message.

BMessageSH_CMH_StartUpPalletLocation

SH informs the CMH of the location of a new pallet on the system. The message payload is the PH assigned to the pallet, the OH in whose sector the pallet is in, and the OH's queue number it is in.

BMessageSH_OH_ConveyerStart

SH informs the OH that is must change from the "Passive" state to the "Stop and Check" state. In this state, the conveyer is active and moving.

BMessageSH_OH_DisableLLC

SH informs an OH that is must disable Low Level Communication, which means it must neither send nor receive messages from the PLC it is assigned to.

BMessageSH_OH_EmergencyStop

SH informs an OH that it must immediately stop all operations by going into the "Passive" state.

BMessageSH_OH_EnableLLC

SH informs an OH that is must enable Low Level Communication, which means it must proceed to send and receive messages to and from the PLC it is assigned to.

BMessageSH_PH_ContinueTask

SH informs a PH that it must move the pallet it is assigned to a new location. The payload is the node location it must move to.

BMessageSH_PH_NewTaskUserDefinedLocation

Message from SH to PH telling the PH where its assigned pallet is located, and what the destination is. The payload is the start location, the destination location, the instance of the CMH the PH needs to communicate with, the OH address array and the PLC setup array.

BMessageSH_SH_HolonInfo

SH sends a message to itself from its startup to its main sequence containing information used in the startup sequence which the SH will also need in its main program.

APPENDIX B: RODRIGUEZ INTERVIEW

The laboratory technician, Reynaldo Rodriguez, was present for the physical setup and reconfiguration of Le Roux's (2013) project and this project. He was interviewed to offer his insight as to whether the reconfigurability of the conveyer had been improved or not. The following questions were asked.

1. What are the advantages and disadvantages to the new distributed control approach? In particular, in terms of:
    a. Time taken to modify the conveyer setup.
    b. Ease in modifying the conveyer setup.
2. In your opinion, is the extra cost and extra initial setup time worth the reconfigurability benefits?

His reply is as follows:

The Bosch-Rexroth TS 2plus conveyor was designed with modularity in mind so moving the hardware components which form part of the conveyor system is not a major problem. However, since the configuration of a station does not change, all wiring and tubing (within the station) remain the same length and follow similar routing. As a result, the main advantages are derived from the reduced tubing and wiring efforts. From this point of view, a distributed control approach is far easier and quicker to modify with only power sources and one network connection needing to be rerouted for each station. In contrast, with the centralised approach, all cables/tubes for each sensor and actuator must be re-laid incurring extra costs since often sections of cable and tube are not of a suitable length to be reused. Further, while reconfiguring centralised control systems with industrial bus systems, there is likely to be a need to get splices or splitters since moving the main bus cable route can often prove too difficult or even impossible. Under certain circumstances, the fact that a station can be individually taken offline (switched off and isolated) can mean that as long as there is redundancy, maintenance or repair can take place without stopping production.

On the other hand, for distributed systems, expensive interfaces would have to be added to each station (for example, the ability to interface with Profibus slaves). With the centralised control approach only one master is required and the extra cost may limit technology options for large distributed control systems. Another related point to consider is what type of communication is required between the central controller and the distributed controllers. In certain cases a simple Ethernet connection may suffice and be cheap to implement but there may be cases where a station may require real time communication with the main controller which can very quickly drive up the cost of the system. The requirement that each station has its own electrical control box can represent a significant expense, depending on the number of stations the system comprises. Volumetrically, one large electrical cabinet will be much cheaper than many smaller ones adding up to the same useful space. In general, this principle extends to most things within the control

cabinet (power supplies in particular). With each control cabinet come extra components that would not be needed otherwise, such as isolators, breakers, buttons, etc.

Given the above advantages and disadvantages, the answer to the question of whether the extra cost and extra initial setup time is worth the reconfigurability benefits of a distributed control approach depends to a large extent on two factors: reconfigurability frequency and system size, but the type of technology used can also have an impact. Because of this, I do not think it is possible to say that a distributed control approach is (or not) worth the extra cost or time of initial set up. This is a question that needs to be answered on a per-project basis, analysing the individual system requirements. A more useful approach to this question would be to design a standardised analysis that concentrates on evaluating key aspects of the system (as partly listed in the above comments) in order to produce an objective result that can help make an informed decision. For our system, it works very well…

APPENDIX C: CONVEYER SETUP 2 DIAGRAM AND DATA ARRAYS

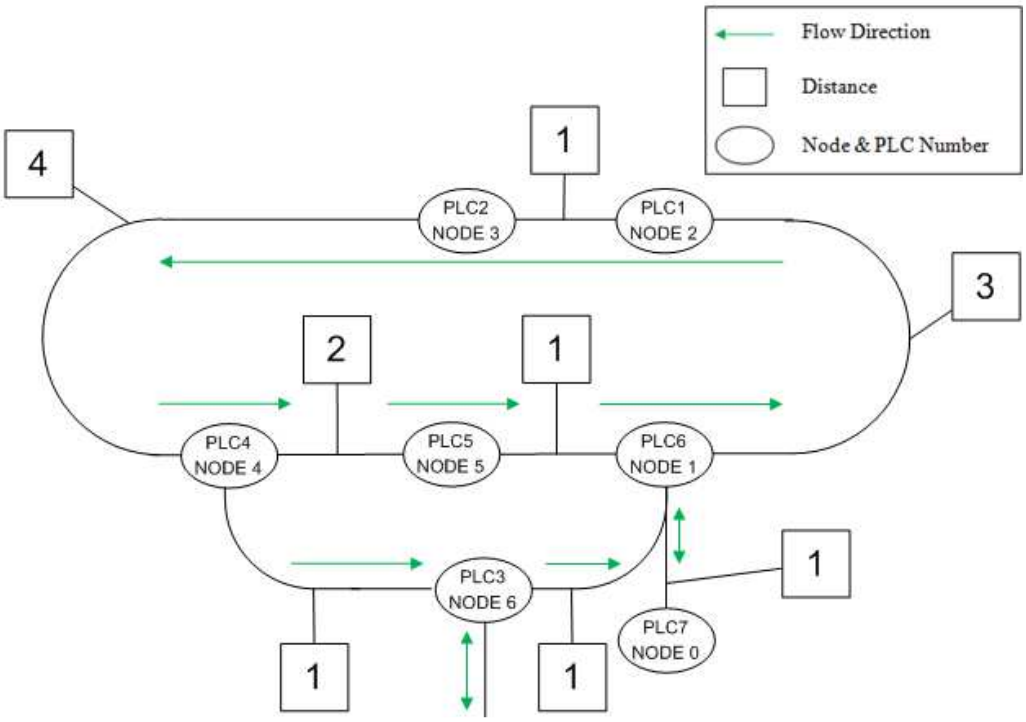


Figure 29: Conveyer setup 2 diagram

Table 14: Conveyer map matrix 2

		Destination Node Number						
		0	**1**	**2**	**3**	**4**	**5**	**6**
Start Node Number	**0**		1					
	1	1		3				
	2				1			
	3					3		
	4						2	1
	5		3			2		
	6		1					

Table 15: PLC mapping matrix 2

	Destination Node Number							
Start Node Number		**0**	**1**	**2**	**3**	**4**	**5**	**6**
	0		6					5
	1	7		1				5
	2				2			
	3					4		
	4						5	3
	5		6			4		
	6		6					

Table 16: Direction number array 2

	Instruction Number		
Module Type		**0**	**1**
	DU	2	0
	DU&PM	5	6

Table 17: PLC feed to node array 2

	Outgoing Sector Number		
Operational Holon ID Number		**0**	**1**
	1	3	
	2	4	
	3	1	
	4	5	6
	5	1	0
	6	2	6
	7	1	

Table 18: Incoming queue array 2

	Incoming Sector Number			
Operational Holon ID Number		0	1	2
	1	One_Two		
	2	Two_Three		
	3	Four_Six		
	4	Three_Four		
	5	Four_Five		
	6	Five_One	Zero_One	Six_One
	7	One_Zero		

Table 19: Outgoing queue array 2

	Outgoing Sector Number		
Operational Holon ID Number		0	1
	1	Two_Three	
	2	Three_Four	
	3	Six_One	
	4	Four_Five	Four_Six
	5	Five_One	
	6	One_Two	One_Zero
	7	Zero_One	

Table 20: Outgoing sector array 2

	Outgoing Sector Number		
Operational Holon ID Number		0	1
	1	0	
	2	0	
	3	2	
	4	0	0
	5	0	
	6	0	0
	7	1	

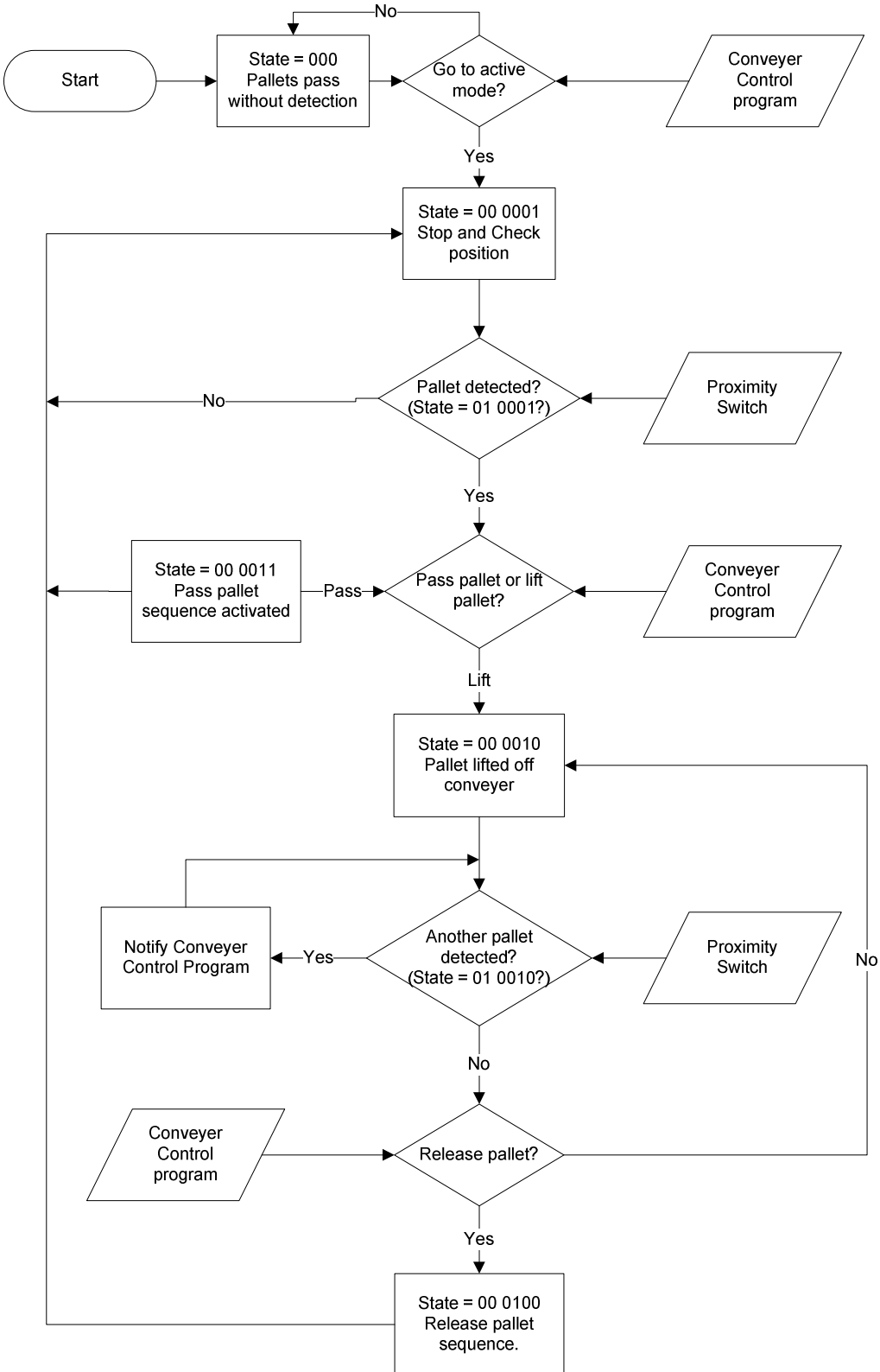
Table 21: Outgoing connect array 2

		Outgoing Sector Number	
Operational Holon ID Number		**0**	**1**
	1	2	
	2	4	
	3	6	
	4	5	3
	5	6	
	6	1	7
	7	6	

APPENDIX D: DETAILED STATE DIAGRAMS

In the state diagrams, the diamond shape indicates a decision which needs to be made. The square box is a process or state. The slanted square or rhombus shape is an information source. The states are indicated in all boxes. Note that the states are in binary format.

Note that in the case where a module was modified for the second conveyer setup (refer to Section **Error! Reference source not found.**) the modified state diagram is shown directly after the original state diagram.



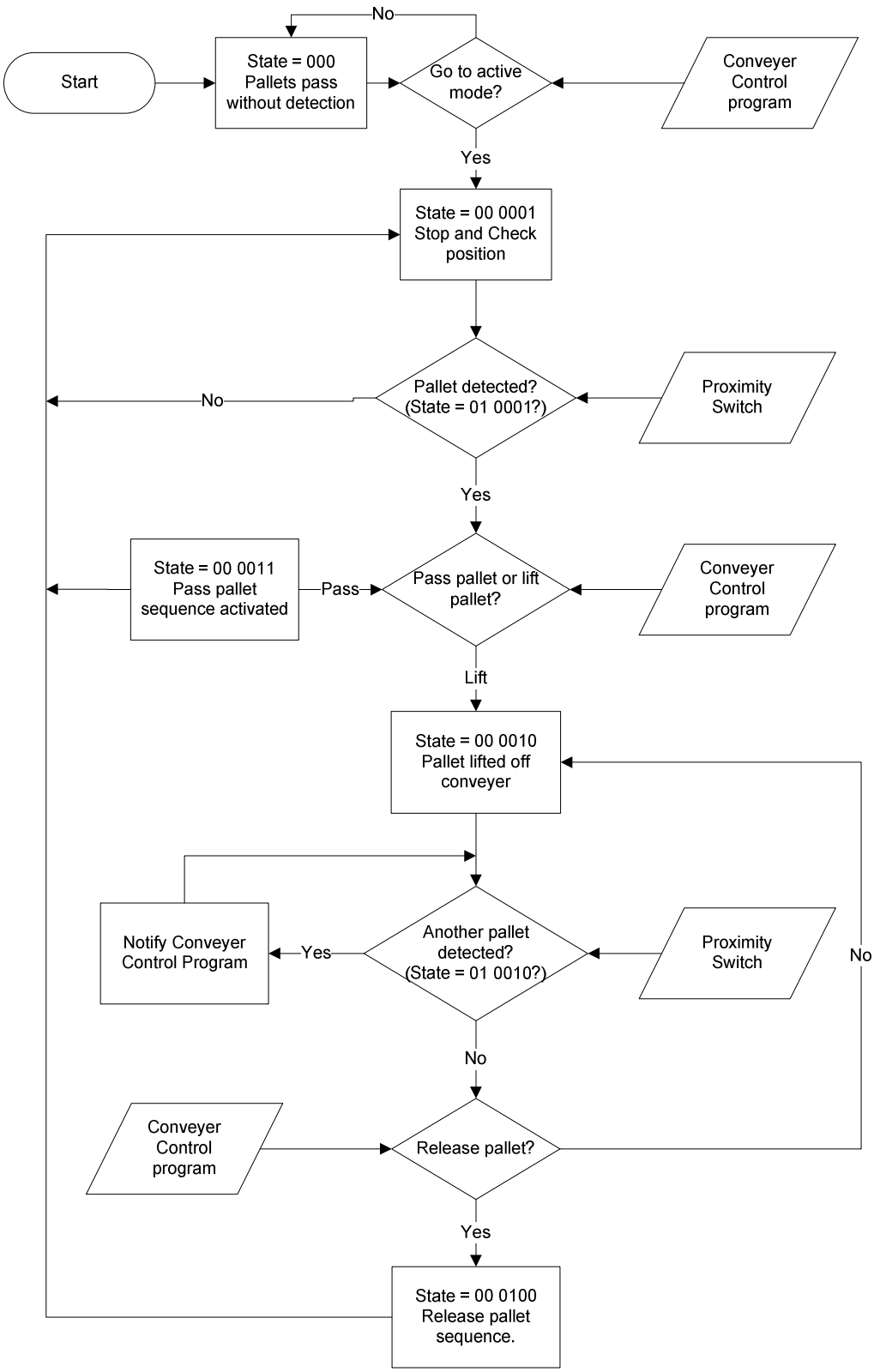


Figure 30: Lifting unit state diagram

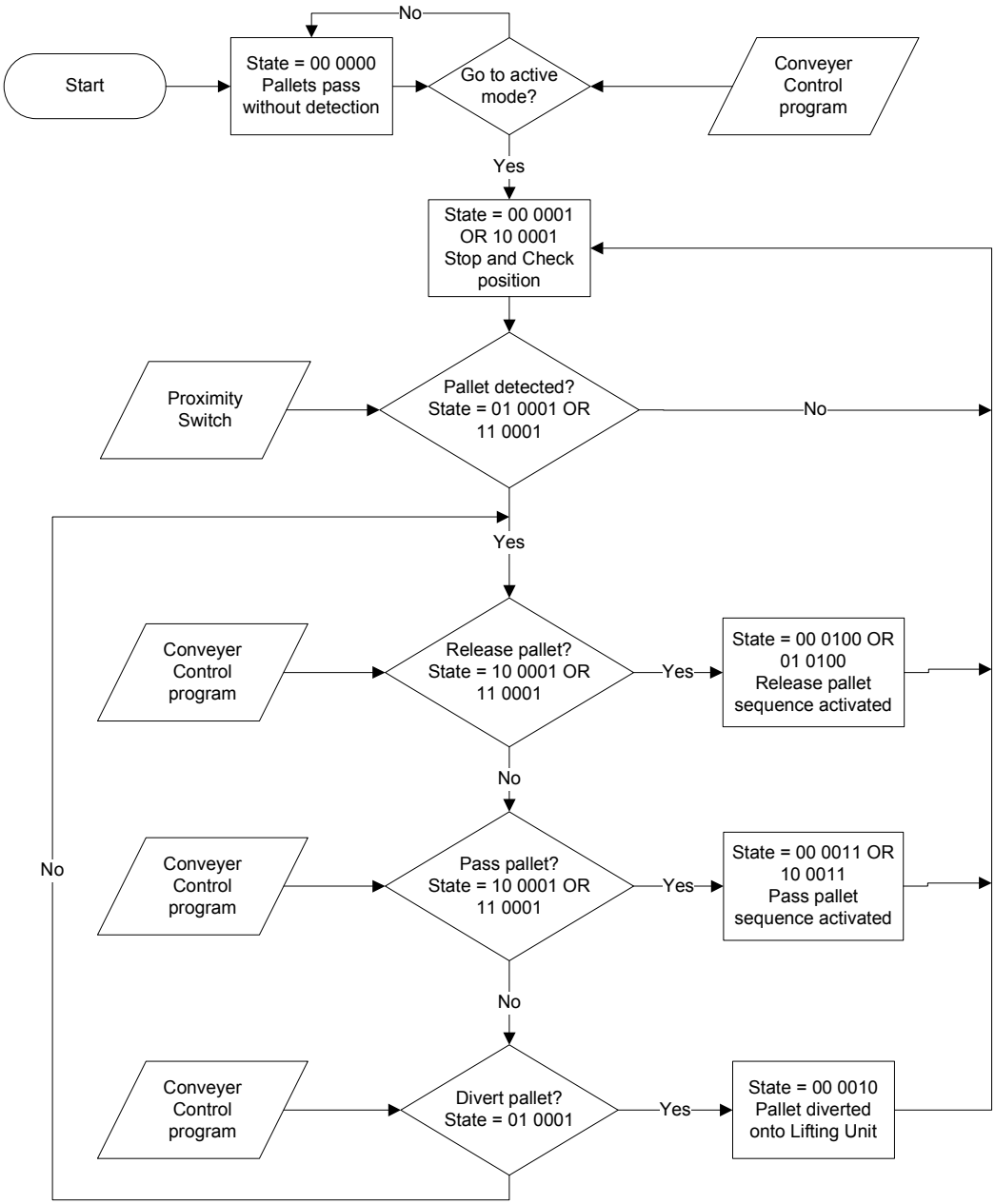


Figure 31: Lifting unit with transverse conveyer state diagram

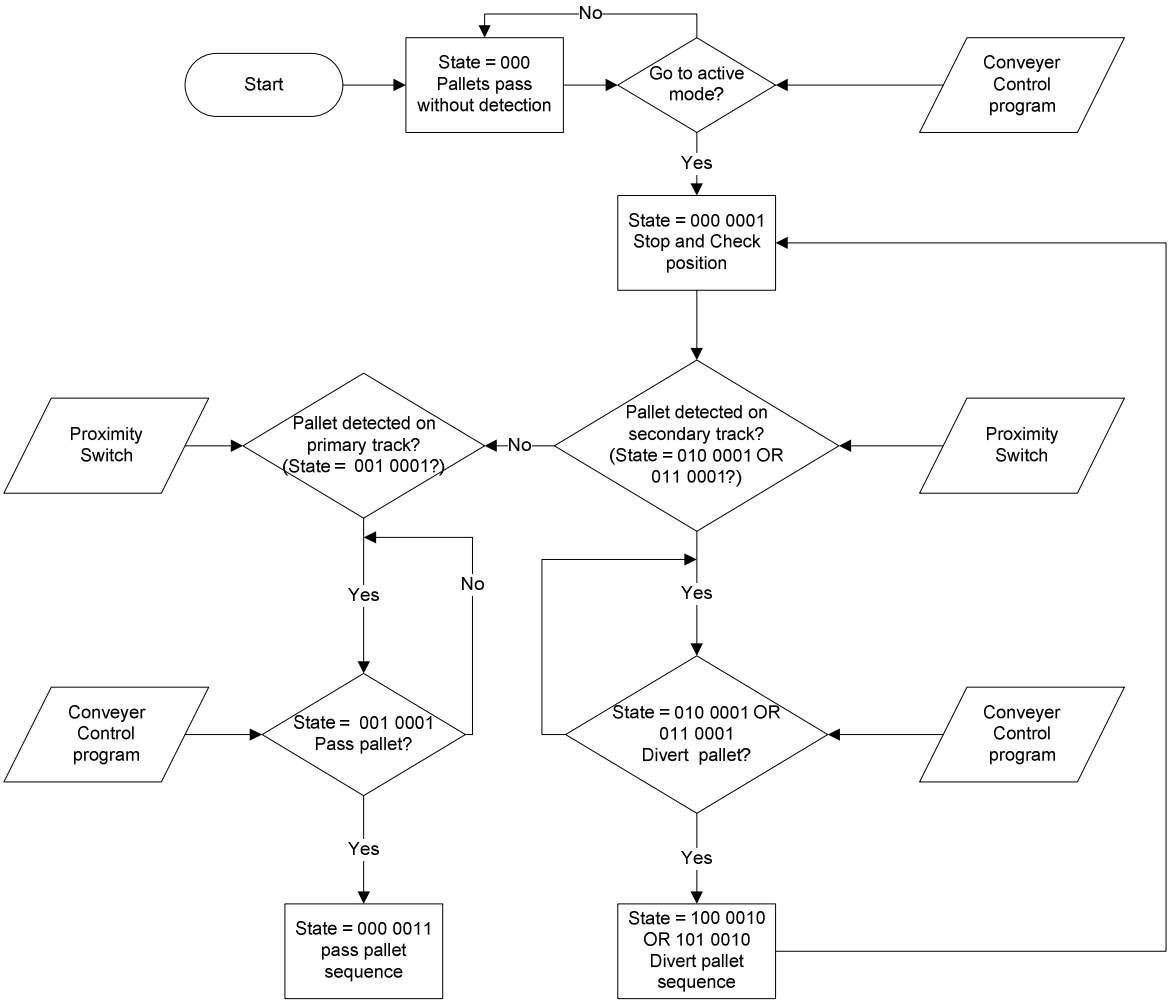


Figure 32: Divert unit state diagram

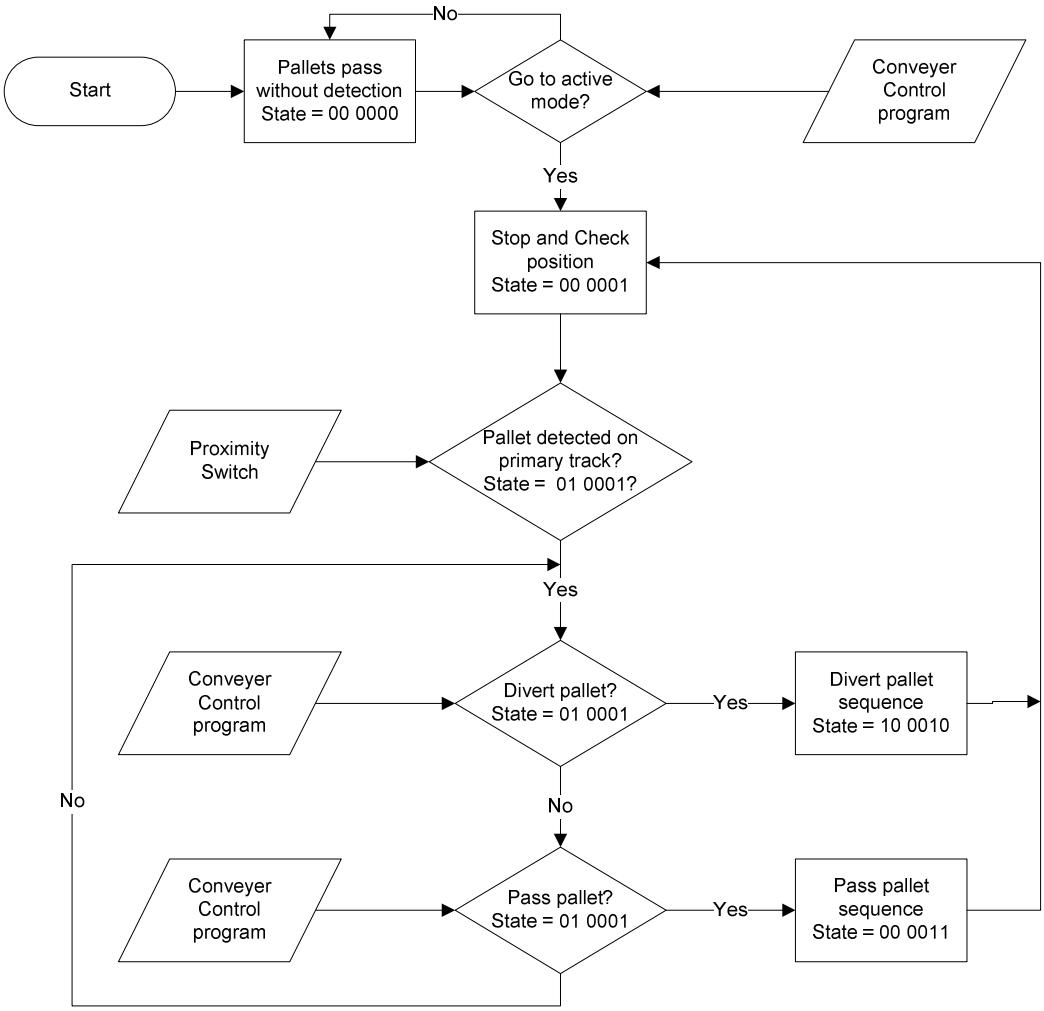


Figure 33: Divert unit state diagram 2

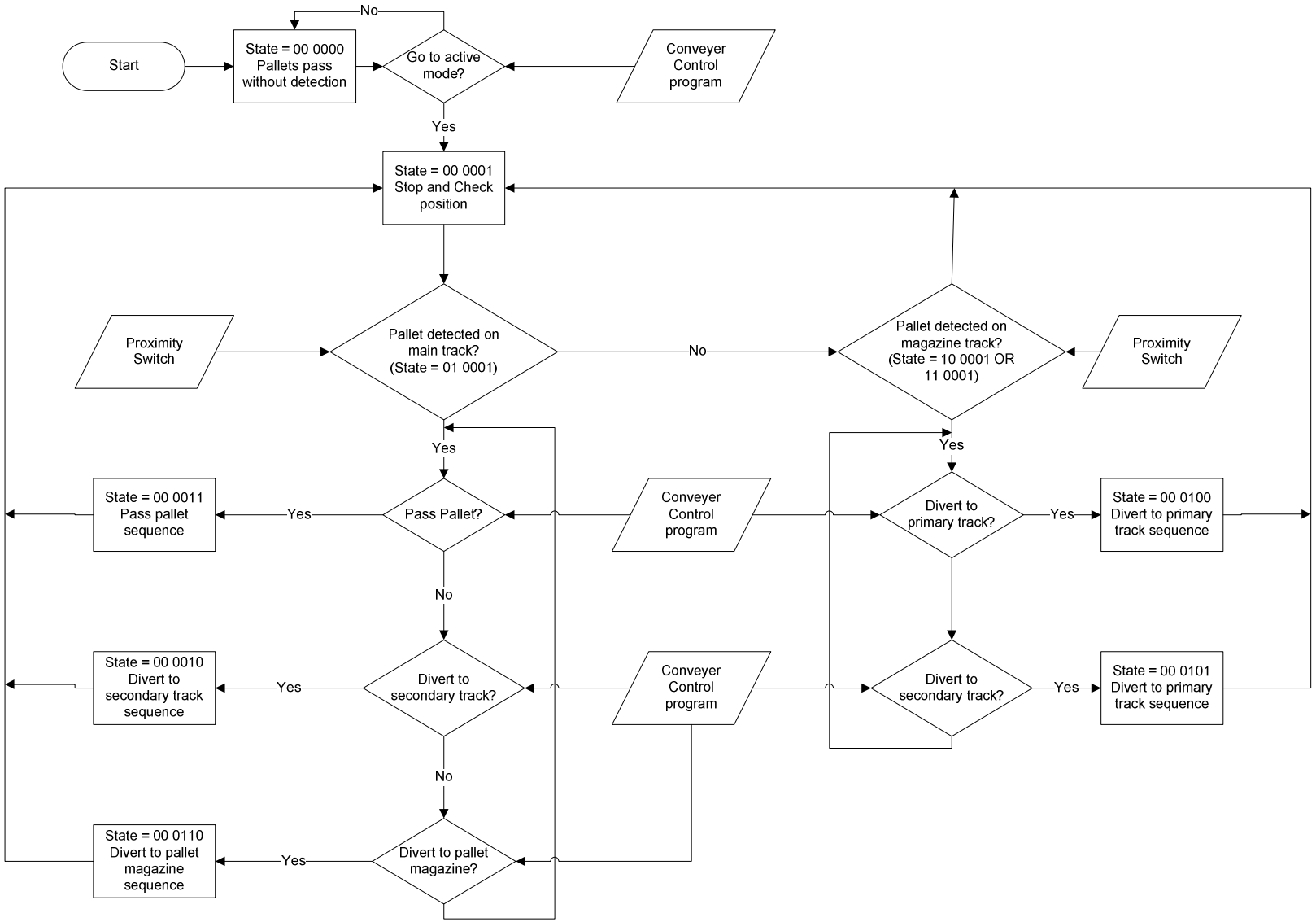


Figure 34: Divert unit with pallet magazine state diagram

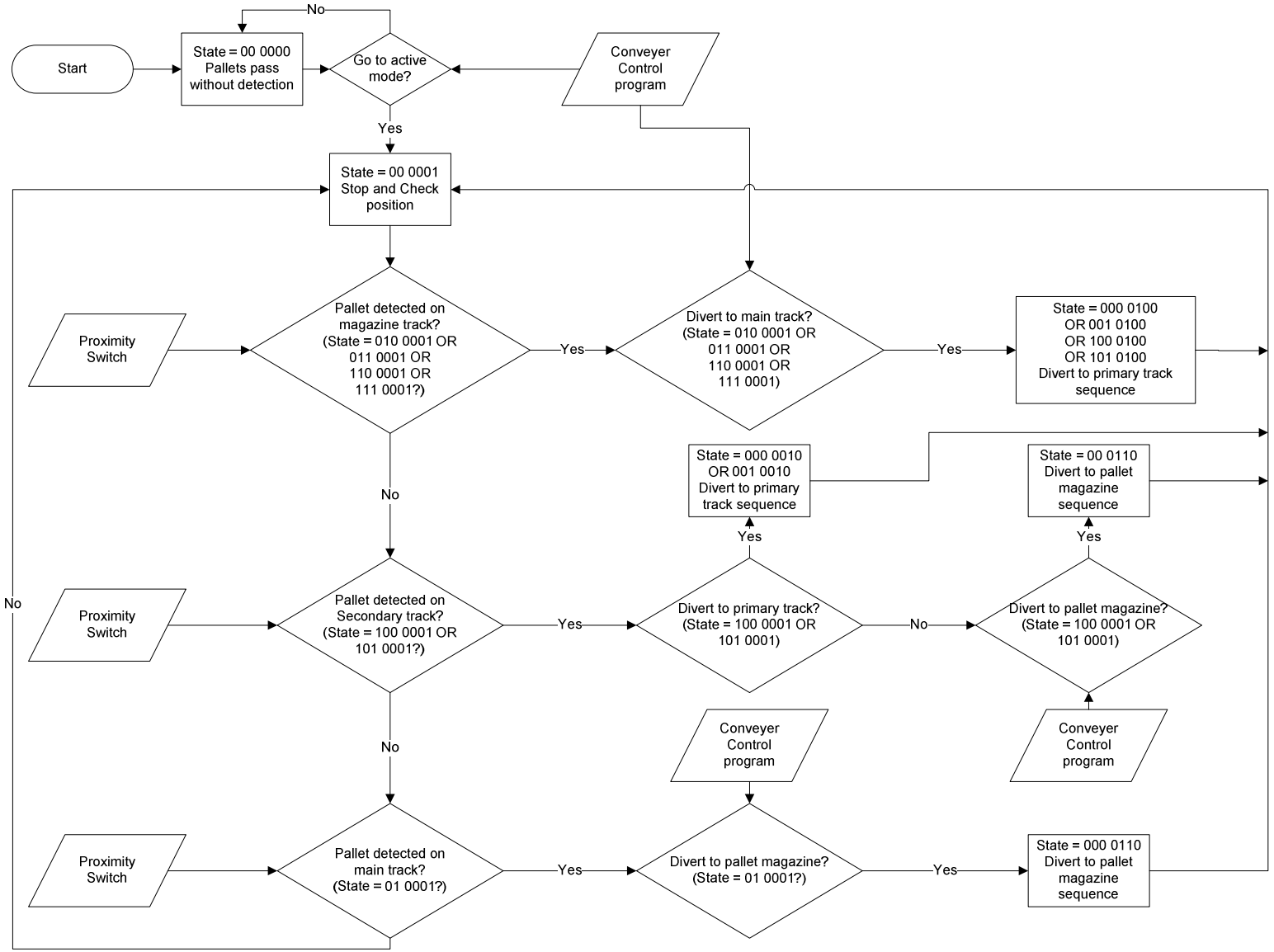


Figure 35: Divert unit with pallet magazine state diagram 2

APPENDIXE: SAMPLE PLC PROGRAMS

E1: Lifting Unit PLC Program

```
CASE BYTE_TO_INT(IN:= %MB0) OF

        // Pass all pallets
0:      S0 := FALSE;
        S1 := FALSE;
        LU := FALSE;

        T0_On := FALSE;
        T2_On := FALSE;
        T3_On := FALSE;

        Motor_On := FALSE;

        Trigger := FALSE;

        //  STOP AND check pallets OR release pallet
1:      S0 := TRUE;
        S1 := FALSE;
        LU := FALSE;

        T0_On := FALSE;
        T2_On := FALSE;
        T3_On := FALSE;

        Trigger := FALSE;

        Motor_On := TRUE;

        // Lift Pallets
2:      S0 := FALSE;
        S1 := TRUE;
        LU := FALSE;

        Motor_On := TRUE;

        T0_On := TRUE;

        T2_On := FALSE;
        T3_On := FALSE;

        IF T0_Q = TRUE THEN
                S0 := TRUE;
        END_IF;
```

```
                IF T1_Q = TRUE THEN
                        LU := TRUE;
                END_IF;

                // Pass pallets
        3:      S0 := FALSE;
                S1 := FALSE;
                T0_On := FALSE;
                T2_On := TRUE;

                IF T2_Q = TRUE THEN
                        Trigger := TRUE;
                END_IF;

                Motor_On := TRUE;

        4:      // Release pallets
                S0 := TRUE;
                S1 := FALSE;
                LU := FALSE;

                T0_On := FALSE;
                T2_On := FALSE;

                T3_On := TRUE;

                IF T3_Q = TRUE THEN
                        Trigger := TRUE;
                END_IF;

END_CASE;

TON_0(IN:=(T0_On), PT:=(T#0.5s), Q=>(T0_Q));
TON_1(IN:=(T0_Q), PT:=(T#1.5s), Q=>(T1_Q));
TON_2(IN:=(T2_On), PT:=(T#0.5s), Q=>(T2_Q));
TON_3(IN:=(T3_On), PT:=(T#3s), Q=>(T3_Q));

LU_Contact := LU;
S0_Contact := NOT S0;
S1_Contact := NOT S1;
Motor_Contact := Motor_On;

Send_Byte.4 := I0;
%MB1 := Send_Byte OR %MB0;

IF Trigger = TRUE THEN
        %MB0 := 2#1;
```

```
END_IF;
```

E2: Divert Unit with Pallet Magazine Module PLC Program

```
CASE BYTE_TO_INT(IN:= %MB0) OF
        0: // Let all pallets pass
                S0 := FALSE;
                S1 := TRUE;
                TC0_0 := TRUE;
                TC0_2 := FALSE;
                TC1_2 := FALSE;

                T0_On := FALSE;
                T3_On := FALSE;
                T4_On := FALSE;
                T5_On := FALSE;
                T6_On := FALSE;

                Motor_On := FALSE;
                Direction := FALSE;
                Motor_Main := FALSE;

                Trigger := FALSE;

        1:      //STOP AND check pallets
                S0 := TRUE;
                S1 := TRUE;
                TC0_0 := TRUE;
                TC0_2 := FALSE;
                TC1_2 := FALSE;

                T0_On := FALSE;
                T3_On := FALSE;
                T4_On := FALSE;
                T5_On := FALSE;
                T6_On := FALSE;

                Motor_On := FALSE;
                Direction := FALSE;
                Motor_Main := TRUE;

                Trigger := FALSE;

        2: // Transfer from main track TO parallel track
                S0 := FALSE;
                S1 := TRUE;
                TC0_0 := FALSE;
                TC0_2 := FALSE;
```

```
        TC1_2 := FALSE;

        Motor_On := FALSE;
        Direction := FALSE;
        Motor_Main := TRUE;

        T0_On := TRUE;

        IF T0_Q = TRUE THEN
                S0 := TRUE;
        END_IF;

        IF T1_Q = TRUE THEN
                TC0_2 := TRUE;
                TC1_2 := TRUE;
                Motor_On := FALSE;
                Direction := TRUE;
        END_IF;

        IF T2_Q = TRUE THEN

                Trigger := TRUE;

        END_IF;

3:      // Pass pallet
        S0 := FALSE;
        S1 := TRUE;
        TC0_0 := TRUE;
        TC0_2 := FALSE;
        TC1_2 := FALSE;

        Motor_On := FALSE;
        Direction := FALSE;
        Motor_Main := TRUE;

        T0_On := FALSE;

        T3_On := TRUE;
        IF T3_Q = TRUE THEN

                Trigger := TRUE;

                S0 := TRUE;
        END_IF;

4:      // Transfer from magazine TO main track
        S0 := TRUE;
```

```
S1 := FALSE;
TC0_0 := FALSE;
TC0_2 := TRUE;
TC1_2 := TRUE;

T0_On := FALSE;
T3_On := FALSE;

Motor_On := TRUE;
Direction := FALSE;
Motor_Main := TRUE;

IF RPS_0 = TRUE THEN

        T5_On := TRUE;

END_IF;

IF T5_Q = TRUE THEN
        Trigger := TRUE;
END_IF;
```

5: // Transfer from Magazine TO secondary track

```
S0 := TRUE;
S1 := Trigger_2;
TC0_0 := FALSE;
TC0_2 := TRUE;
TC1_2 := TRUE;

T0_On := FALSE;
T3_On := FALSE;

Motor_On := NOT Trigger_2;
Direction := Trigger_2;
Motor_Main := TRUE;

IF RPS_0 = TRUE THEN
        T4_On := TRUE;
        Trigger_2 := TRUE;
END_IF;


IF T4_Q = TRUE THEN
        Trigger_2 := FALSE;
        Trigger := TRUE;
END_IF;
```

6: // Transfer from main track TO magazine

```
                S0 := FALSE;
                S1 := FALSE;
                TC0_0 := FALSE;
                TC0_2 := FALSE;
                TC1_2 := FALSE;

                Motor_On := FALSE;
                Direction := FALSE;
                Motor_Main := TRUE;

                T6_On := TRUE;

                IF T6_Q = TRUE THEN
                        S0 := TRUE;
                END_IF;

                IF T7_Q = TRUE THEN
                        TC0_2 := TRUE;
                        TC1_2 := TRUE;
                        Motor_On := FALSE;
                        Direction := TRUE;
                END_IF;

                //IF T8_Q = TRUE THEN
                IF PS_1 = TRUE THEN
                        Trigger := TRUE;
                END_IF;

END_CASE;

S0_Contact := NOT S0;
S1_Contact := NOT S1;
TC0_0_Contact := TC0_0;
TC0_2_Contact := TC0_2;
TC1_2_Contact := TC1_2;

Motor_On_Contact := Motor_On;
Direction_Contact := Direction;
Motor_Main_Contact := Motor_Main;

PS_0 :=PS_0_Contact;
PS_1 := NOT PS_1_Contact;
RPS_0 := RPS_0_Contact;

Send_byte.4 := PS_0;
Send_byte.5 := PS_1;
Send_byte.6 := RPS_0;
```

```
%MB1 := Send_byte OR %MB0;

TON_0(IN:=(T0_On), PT:=(T#0.75s), Q=>(T0_Q));          // STOP gate timer
TON_1(IN:=(T0_Q), PT:=(T#1.5s), Q=>(T1_Q));            // Pallet mount timer
TON_2(IN:=(T1_Q), PT:=(T#2.75s), Q=>(T2_Q));
TON_3(IN:=(T3_On), PT:=(T#0.75s), Q=>(T3_Q));
TON_4(IN:=(T4_On), PT:=(T#2s), Q=>(T4_Q));
TON_5(IN:=(T5_On), PT:=(T#1s), Q=>(T5_Q));
TON_6(IN:=(T6_On), PT:=(T#0.75s), Q=>(T6_Q));
TON_7(IN:=(T6_Q), PT:=(T#1.5s), Q=>(T7_Q));
TON_8(IN:=(T7_Q), PT:=(T#3.5s), Q=>(T8_Q));

IF Trigger = TRUE THEN
        %MB0 := 2#1;
END_IF;
```