DEVELOPMENT OF A RECONFIGURABLE ASSEMBLY SYSTEM WITH AN INTEGRATED INFORMATION MANAGEMENT SYSTEM

LYLE CHRISTOPHER SMITH

Dissertation submitted in fulfilment of the requirements for the

MAGISTER TECHNOLOGIAE:

ENGINEERING ELECTRICAL

In the

Department of Electrical, Electronics and Computer Engineering

of the

Faculty of Engineering and Information Technology

at the

Central University of Technology, Free State

Supervisor: Prof HJ Vermaak, PhD

DECLARATION

I, LYLE CHRISTOPHER SMITH, identity number [redacted] and student number 205004148, do hereby declare that this research project which has been submitted to the Central University of Technology, Free State, for the Degree Magister Technologiae: Engineering Electrical, is my own independent work and complies with the Code of Academic Integrity, as well as other relevant policies, procedures, rules and regulations of the Central University of Technology, Free State, and has not been submitted before by any person in fulfilment (or partial fulfilment) of the requirements for the attainment of any qualifications.

Signature

10 February 2014

Date

ACKNOWLEDGEMENTS

I would like to thank the following people who contributed towards the completion of the dissertation:

- I would like to thank my parents for always believing in me and encouraging me and always making sure to check on my progress.
- Prof H Vermaak for his guidance, support and help with the final documentation.
- The Central University of Technology, Free State, for the provision of research facilities and financial support.
- To all my colleagues and individuals not mentioned here who helped or supported me in any way.

ABSTRACT

This dissertation evaluates the software and hardware components used to develop a Reconfigurable Assembly System with an Integrated Information Management System. The assembly system consists of a modular Cartesian robot and vision system. The research focuses on the reconfigurability, modularity, scalability and flexibility that can be achieved in terms of the software and hardware components used within the system.

The assembly system can be divided into high-level control and low-level control components. All information related to the product, Cartesian positioning and processes to follow resides in the Information Management System. The Information Management System is the high-level component and consists of a database, web services and low-level control drivers. The high-level system responds to the data received from the low-level systems and determines the next process to take place. The low-level systems consist of the PLC (Programmable Logic Controller) and the vision system. The PLC controls the Cartesian robot's motor controllers and handles all events raised by field devices (e g. sensors or push buttons). The vision system contains a number of pre-loaded inspections used to identify barcodes and parts, obtain positioning data and verify the products' build quality. The Cartesian robot's positioning data and the vision system's inspections are controlled by the Information Management System.

The results showed that the high-level control software components are able to add more modularity and reconfigurability to the system, as it can easily adapt to changes in the product. The high-level control components also have the ability to be reconfigured while the assembly system is online without affecting the assembly system. The low-level control system is better suited to handling the control of motor controllers, field devices and vision inspections over an industrial network.

ABSTRAK

Hierdie verhandeling evalueer die sagteware- en hardewarekomponente wat gebruik is om ’n Herkonfigureerbare Monteringstelsel met ’n Geïntegreerde Inligtingbeheerstelsel (“Integrated Information Mangement System”) te ontwikkel. Die monteringstelsel bestaan uit ’n modulêre Cartesiese robot en visiestelsel. Die navorsing fokus op die herkonfigureerbaarheid, modulariteit, skaalbaarheid en buigsaamheid wat bereik kan word gegewe die sagteware- en hardewarekomponente wat in die stelsel gebruik word.

Die monteringstelsel bestaan uit hoëvlakbeheer- en laevlakbeheerkomponente. Alle inligting oor die produk, Cartesiese posisie en prosesse wat volg, is in die Inligtingsbeheerstelsel gesetel. Die Inligtingbeheerstelsel verteenwoordig die hoë-vlak komponent en bestaan uit ’n databasis, webdienste en laevlakbeheermeganismes. Die hoë-vlakstelsel reageer op die data wat van die laevlakstelsels af ontvang word en bepaal die volgende proses wat moet plaasvind. Die laevlakstelsels bestaan uit ’n PLC (“Programmable Logic Controller”) en ’n visiestelsel. Die PLC beheer die Cartesiese robot se motorkontroleerders en hanteer alle voorvalle veroorsaak deur veldtoestelle (bv. sensors of drukknoppies). Die visiestelsel bevat ’n aantal voorafgelaaide inspeksies wat gebruik word om strepieskodes en onderdele te identifiseer, posisiedata te verkry en produkboukwaliteit te verifieer. Die Cartesiese robot se posisiedata en die visiestelselinspeksies word deur die Inligtingbeheerstelsel gekontroleer.

Die resultate toon dat die hoëvlakbeheersagtewarekomponente in staat is om meer modulariteit en herkonfigureerbaarheid tot die stelsel toe te voeg, omdat dit maklik by veranderinge in die produk kan aanpas. Die hoëvlakbeheerkomponente kan verder ook herkonfigureer word terwyl die monteringstelsel aanlyn is, sonder om die monteringstelsel te beïnvloed. Die laevlakbeheerstelsel is meer gepas vir die hantering van motorkontroleerders, veldtoestelle en visie-inspeksies in ’n industriële netwerk.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

PLC – Programmable Logic Controller

RAS – Reconfigurable Assembly System

FMS – Flexible Manufacturing System

DMS – Dedicated Manufacturing System

RMS – Reconfigurable Manufacturing System

WLAN – Wireless Local Area Network

AGV – Automated Guided Vehicle

PC – Personal Computer

RFID – Radio Frequency Identification

FOV – Field of View

ID – Identification

IP – Internet Protocol

TCP – Transmission Control Protocol

TCP/IP – Transmission Control Protocol/Internet Protocol

UDP – User Datagram Protocol

FCT – Festo Configuration Tool

FHPP – Festo Handling and Positioning Profile

FPC – Festo Parameter Channel

VBAI – Vision Builder for Automated Inspection

ERP – Enterprise Resource Planning

GUI – Graphical User Interface

OCR – Object Character Recognition

HMI – Human Machine Interface

SCADA – Supervisory, Control and Data Acquisition

IIS – Internet Information Services

Chapter 1: An Introduction to the Study

1.1 Introduction

Manufacturing companies are faced with a problem in terms of which they need to adjust to market changes rapidly. These changes could include the introduction of a new product, changes in demand or changes in batch orders. The driving factor behind the design and development of flexible assembly systems is economics, with the world market demanding greater product variety, consistent high quality, competitively priced products and rapid new product introduction [1].

Traditional manufacturing ideas resort to using manual assembly when producing complex parts instead of dedicated manufacturing systems. This is justified by the high cost of installing a dedicated manufacturing line and also the inability of dedicated lines to adjust to the introduction of a new product or any changes to the existing products. Dedicated assembly lines cannot easily adjust to dimensional changes of the families of parts as well as the additional handling requirements.

Automated assembly systems must be able to compete at all levels of production to prove feasible. In this respect, an automated assembly system must be able to produce at a high volume, provide a rapid part change-over and comprise the ability to ramp-up to full production capability over a short period of time. The system's structure and capabilities should hold a certain degree of flexibility. The need for flexible systems is the driver for developing an assembly system that has reconfigurable abilities in terms of the hardware and the software.

1.2 Problem Statement

Dedicated production lines cannot easily adjust to facilitate the assembly of variations of families of parts as well as the added manipulation requirements of such variations. Manual

assembly has proved too costly to some South African companies because of the high cost and/or low productivity of labour. By developing a Reconfigurable Assembly System, these industries can retain the assembly operations in South Africa and thereby protect the jobs of South African workers.

1.3 Research Goals and Objectives

1.3.1 Hypothesis

A Reconfigurable Assembly System (RAS), with an Integrated Information Management System, will be produced that allows for a variety of similar product types to be assembled by dynamically changing the system's physical structure and system control software.

1.3.2 Specific Objectives

The objectives of this study are as follows:

- To develop a RAS that is able to assemble various products on the same line. The product will consist of a minimum of 3 components.
- To utilise a vision system to aid the RAS in the assembly and inspection processes of the various products.
- To develop software that allows the RAS to communicate with the Information Management System and Windows Mobile devices.

1.4 Research Methodology

A number of aspects had to be taken into account when designing a Reconfigurable Assembly System (RAS). The physical design of the RAS as well as the software, hardware and communication methods used to control the RAS are important factors. Another

important factor that needed to be considered is that the RAS needs to be able to run independently of the main system. These factors formed the requirements needed to create a RAS.

A Reconfigurable Assembly System with an Integrated Information Management System was produced by means of the following:

- The selection of a suitable Programmable Logic Controller (PLC) to control the RAS stepper motor controllers and receive position feedback from the stepper motors over a Profibus network.
- A PLC driver was written in C# that allows the Information Management System and the PLC to exchange data over Ethernet.
- A SQL database was created, for the Information Management System, that contains information related to all available products; RAS information and product build traceability information.
- A vision system was programmed to scan barcodes, locate parts, inspect specific objects, switch to different inspections and verify that a product was built correctly.
- A driver was written in C# that allows the vision system and the Information Management System to exchange data over Ethernet.
- An Information Management System was programmed to have the following functions:
    - Create, Edit or delete a product.
    - Retrieve job information for a RAS.
    - Calibrate the vision system and the Cartesian system.
    - Display RAS status information.
- A Web Service was programmed in C# that exposes commands to computers and Windows Mobile devices that enables them to view the Information Management System data.
- A Windows Mobile application and a desktop application were programmed in C# that subscribes to the web service and displays the requested information.

- A Windows Mobile application was programmed in C# that allows the Windows Mobile device to control some RAS functions (e.g. start, stop, reset, manual control, etc.) over Wireless Local Area Networks (WLAN).

1.5 Layout of Dissertation

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

Chapter 2: This chapter provides insight into the different manufacturing systems currently used and characteristics of Reconfigurable Assembly Systems.

Chapter 3: This chapter discusses the development and implementation of the Reconfigurable Assembly System with regard to the hardware, software, communication and system integration.

Chapter 4: This chapter discusses the test methods and analysis used for the vision system and the Cartesian system.

Chapter 5: This chapter provides a conclusion to the study and recommendations for future research.

1.6 Chapter References

[1] Edmondson, N. F. and Redford, A. H., 2002, *Generic flexible assembly system design*, Assembly Automation, Vol 22, no 2, pages 139-152.

Chapter 2: Literature Review

The research presented below provides insight into the various manufacturing systems currently used with an emphasis on Reconfigurable Assembly Systems (RAS). The literature also describes the various subsystems that are required to develop a RAS.

2.1. Manufacturing Systems

There are many different manufacturing systems used in the manufacturing industry today. Although the RAS is the core focus of this study, it is necessary to understand the other manufacturing systems and how the RAS differs from them. There are two traditional types of manufacturing systems used in today's manufacturing industry, namely Dedicated Manufacturing Systems (DMS) and Flexible Manufacturing Systems (FMS) [1]. A Reconfigurable Manufacturing System (RMS) represents another type of manufacturing system said to incorporate the best qualities of both the DMS and the FMS [1].

2.1.1. Dedicated Manufacturing Systems

Dedicated Manufacturing Systems (DMS) are designed for the cost-effective production of a specific part at high volumes and good quality using fixed tooling and inexpensive fixed automation. The sole objective of a DMS is to mass produce a product at a low cost. This only proves true when the demand for the product exceeds the supply. The use of DMS's in today's industry is fading away as the current market demands product variety. A dedicated line is designed to produce a specific product and cannot easily adapt without making major changes to the system. Dedicated lines do not usually operate at full capacity as a result of the increasing pressure from global competitors. This causes the DMS to be cost-ineffective and not a viable option as a manufacturing system in the present market.

2.1.2. Flexible Manufacturing Systems

The term Flexible Manufacturing Systems (FMS) is defined by Setchi [2] as a manufacturing system configuration with fixed hardware and fixed, but programmable, software to handle changes in work orders, production schedules, part programs and tooling for several types of parts. ELMaraghy [3] describes a FMS as an integrated system of machine modules and material handling equipment under computer control for the automatic random processing of palletized parts. Koren [4] defines a FMS as a manufacturing system that consists of computer numerically controlled (CNC) machines and other programmable automation and can produce a variety of products on the same system.

A FMS is a flexible system as it is able to produce a variety of products on the same line; however, it is unable to match the high throughput of a DMS. The equipment required to obtain flexibility, results in high equipment price. This, combined with the low throughput rate, causes the cost per product to be relatively high. The design of the FMS focuses on the machine and not the product. The FMS is best suited to produce small sets of products.

2.1.3. Reconfigurable Manufacturing Systems

A Reconfigurable Manufacturing System (RMS) is defined by Mehrabi et al. [5] as a machining system which can be created by incorporating basic process modules – both hardware and software – that can be rearranged or replaced quickly and reliably. This type of system provides customized flexibility for a particular part family and will be open-ended so that it can be improved, upgraded, and reconfigured rather than being replaced. An RMS is designed for rapid change in structure in order to quickly adjust production capacity and functionality, within a part family, in response to changes in market requirements [3].
The objective is to provide exactly the functionality and capacity that is needed, when it is needed.

An RMS has been considered a cost-effective strategy for the manufacturing of product families. An RMS is focused on the product family and not just one product in the case of DMS and the machine in the case of the FMS. This allows the RMS to respond quickly and cost-effectively to product family design changes.

The following characteristics and principles make up an RMS [6]:

- **Customization:** The machine's flexibility is limited to the product family and not just one part.
- **Convertibility:** The ability to transform the functionality of existing machines easily to suit new production requirements.
- **Scalability:** The ability to modify the existing production capacity easily by adding or removing machines and/or changing system components.
- **Modularity:** The machine components must be based on functional units that can be arranged optimally for alternate production schemes.
- **Integrability:** The ability to integrate modules rapidly and precisely by a set of mechanical, informational and control interfaces that facilitate integration and communication.
- **Diagnosability:** The ability to automatically read the current state of a system to detect and diagnose the root causes of product defects and correct operational defects quickly.

Customization, scalability and convertibility are critical reconfiguration characteristics [7]. Modularity, Integrability and diagnosability allow for rapid reconfiguration, but they do not guarantee modifications in production capacity and functionality. These six characteristics of an RMS reduce the time and effort of reconfiguration and thus enhance the system's response time. They also allow the machine to respond quickly to market changes, product changes and machine failures.

The objective of an RMS is to not only combine the high throughput of the DMS with the flexibility of the FMS, but also to react to changes quickly, efficiently and in a cost-effective way.

2.2. Reconfigurable Assembly System

A Reconfigurable Assembly System (RAS) is considered a key component of a larger RMS, with the assembly stage seen as one of the intermediate steps necessary when manufacturing a product which consists of multiple parts. One definition by Yu et al. [8] is that a RAS is an integrated, computer-controlled system of assembly robots, automated guided vehicles (AGV's) and buffers that can be used to assemble a variety of products. The RAS is built of reconfigurable hardware and software and similarly to an RMS, it provides customized flexibility for a particular family of parts.

The increasing need for new methods of automating assembly processes is driven by frequent market changes and the application of just-in-time production techniques, where parts are required to arrive at the work station exactly when they are needed. These market changes include rapid-frequency introduction of new products, changes in product demand and mix and large fluctuations in batch orders, as explained by Koren [9].

The traditional method used by manufactures – which is to use manual assembly for complex parts, instead of special purpose lines - is justified by the high cost of developing a dedicated line and the inability of the dedicated line to accommodate any part variation. Dedicated lines also have the inability to be physically adjusted to suit the dimensional variations of families of parts as well as the added manipulation requirements of such variations.

Any automated assembly system in today's world must be able to compete at all levels of production to prove feasible within the scenario described above. A competitive automated system must be capable of the following: high volume productivity, rapid part changeover and rapid ramp-up time to full-production ability. Such a system must include a certain degree of flexibility with respect to the system structure and system capabilities. This flexibility must not only be in terms of fast changeover between the production of different products, but also fast adaptations to the changing market trend through fast changes of manufacturing systems and the introduction of new products. This gives rise to the concept of systems that comprise reconfigurability.

Reconfigurability is defined by Setchi [2] as the ability to repeatedly change and rearrange the components of a system in a cost-effective way. The prospect of an assembly system possessing the characteristics of reusability, scalability, agility and reconfigurability has resulted in an assembly paradigm known as Reconfigurable Assembly Systems (RAS).

The basic attributes of a reconfigurable assembly system provide a concept which is designed to meet the ever-increasing demands of production. Molina et al. [10] provides the following key attributes applicable to reconfigurable machines:

- Designed to provide the production capacity needed to meet market demands.
- Readily updated with new technologies.
- Rapidly reconfigure to manufacture new products and accommodate changes in production volumes.
- Designed to provide modularity, integratability, convertibility, diagnosability and customization.

Reconfigurable systems can, therefore, be considered to be systems and tools that:

- Give manufactures the ability to adapt to changes in the market quickly.
- Allow different products to be produced with simple and fast reconfiguration.

Reconfigurable systems should have some of the following characteristics:

- They include a computer controlled systems (e.g. PLC, PC) of assembly modules that can be used to assemble a variety of products of the same family type.
- The system can respond dynamically to the changes in product by reconfiguring its physical structure as well as the system control software.
- The production capacity and functionality are not fixed.
- A RAS can be created through the use of reconfigurable hardware components and software components.
- Since a RAS is a particular type of reconfigurable manufacturing system, it will provide a customized flexibility for a particular product family.

The typical modules in a RAS are shown in Figure 2.1

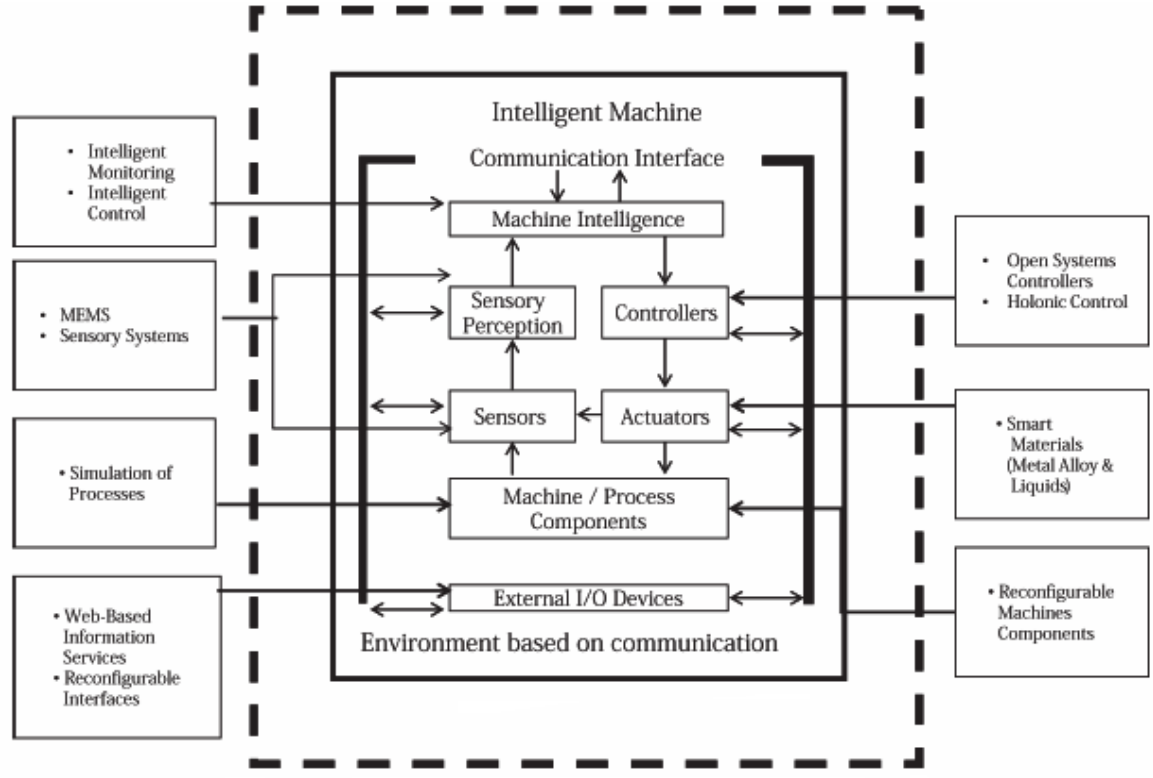


Figure 2.1: Typical modules of a reconfigurable assembly system [11]

The characteristics of a RAS imply that the system is flexible and there are mainly three kinds of flexibility that are required in assembly systems:

- **Type 1: Flexibility for geometrical change:**
  - Type 1 is necessary against changes in size and shape of products. The introduction of manipulators has increased this type of flexibility. Grippers and part feeders still have a small degree of flexibility.
- **Type 2: Flexibility for sequential change**
  - Type 2 is realised by introducing multi-agents in order to accommodate the frequent change in order. Assembly sequences are computed in real time at the assembly line.
- **Type 3: Flexibility for production quantity change:**
  - This type of flexibility proves to be very difficult to achieve. Reconfigurability of assembly systems is a key factor for type 3 to be realised. When a new subsystem such as a manipulator is mounted into an assembly system, the subsystem should be able to work instantaneously.

When looking at the characteristics of a RAS, the following can be summarised from the literature:

- An assembly system often needs to change its configuration of assembly devices such as adding or removing conveyors and part feeders.
- Reconfiguration is required in the event of a breakdown, replacement of a device or even a change of product.
- To decrease the start-up time, the reconfiguration of the entire system should be separated from that of the device. The interface between the device and the control system must be established as easily as possible.

2.3. Conveying Methods

The material flow through an assembly system can be classified into the transport of products or partially assembled products from one assembly station to another and the transport of parts to assembly stations. A key feature of a RAS is a modular conveyor system that can operate asynchronously and be reconfigured to accommodate a large variety of component choices according to the product being assembled [12]. A reconfigurable conveyor allows for the quick rearrangement to alter process flow, adding or bypassing assembly stations according to the desired product. It allows for serial-parallel configurations to balance the assembly line flow to ensure an even throughput.

The most common type of transfer method employed in industry is the conveyor system. A conveyor system is a piece of handling equipment used to move material from one location to another. There are many different types of conveyor systems available that are used according to the various needs and industries. Conveyor systems consist of linear and curved sections that are used to provide a transport path from the start of production to the end of production. These types of systems are usually in a start-to-finish configuration or in a circular configuration. Since these kinds of conveyor systems are serial by design, they do not allow the flow of material to take a different path unless a mechanical device is used to

push the material onto a different path. This kind of system will not be effective when developing a RAS.

Another type of conveyor system used in the transport of material and products is a pallet handling system. A pallet handling system consists of low-friction tracks, rails or guides that allow pallets to be placed on and move independently of one another. This type of system is modular and allows for the easy addition and removal of routes. A pallet system is usually set up in a closed loop system whereby the pallets recirculate in the system. There are a number of advantages to using a pallet system in a reconfigurable environment. These advantages are:

- The pallets serve as assembly fixtures.
- The pallet design allows for one pallet to be used for multiple products.
- A pallet can be used to transport parts and tools to the various assembly stations.
- One pallet is able to hold multiple products at the same time.

These pallets also allow for the integration of Radio Frequency Identification (RFID). An active RFID chip may be used to carry information relating to the product and the progress and state of the product being assembled on the RFID chip. Information is written to these chips using an RFID writer. A passive RFID chip only provides an identification number and thus needs a connection to a database to get the required information relating to the product on the pallet. The use of RFID technology on the pallets allows for various routing plans to be implemented depending on the product to be assembled. This works by having an RFID reader at junction points on the conveyor system. The RFID reader identifies the part and reads the required data to determine the destination location.

An example of a pallet system that allows the use of RFID is the Bosch TS1 transfer pallet system (Figure 2.2). This type of system is modular and allows for various easy additions of alternative routes with little mechanical change. It allows a pallet to be transferred from the main loop onto a subsection where an operation needs to be performed at an assembly cell. RFID read heads are placed at junction points and at workstations to retrieve the data embedded on the RFID chip. The pallet has a removable RFID chip attached to it.

Figure 2.2: Bosch TS1 Pallet Transfer Conveyor

A very simple and cost-effective transfer system is a rotary table. Rotary tables are circular in shape and have a fixed number of stations. The number of stations is directly proportional to the diameter of the rotary table. This type of transfer system is not flexible and hard to reconfigure. If the number of stations needs to be increased then so does the size of the rotary table. This makes the rotary table inefficient. Since rotary tables rotate in a set sequence, the stations must remain fixed relative to one another. There is no flexibility when it comes to rerouting a product on this type of conveyor system. The production rate of this type of conveyor system is determined by the slowest workstation.

Another type of transport system is the use of Autonomous Guided Vehicles (AGVs). The AGVs transport parts, partly assembled products and finished products between workstations whereby they autonomously decide about their particular route. Since AGVs do not need fixed tracks to drive around, they can move in any sequence or order making an AGV transport system very flexible. This type of flexibility is very expensive to implement as AGVs require sophisticated intelligence. This intelligence is used to perform path planning and collision avoidance with other AGVs within the system [13]. An AGV requires its own dedicated power source to be able to operate making it dependent on a charging station for it to be able to be operational.

Transfer systems that are fixed between workstations do not offer the reconfigurability and flexibility associated with a RAS. Although an AGV is a highly flexible transfer system, they

come at a high cost. A cost-effective and flexible solution is seen in a pallet-handling conveyor system.

2.4. Machine Vision

Vision technology deals with images with the objective of manipulating and analysing them in order to improve image quality (contrast), restore image quality (noise reduction), code reading (barcode reading, Object Character Recognition (OCR)) and interpret images (pattern matching, pixel count). Vision technology is used in many different industries for various applications; for example, in biology (counting cells), in construction (thermal analysis) and also in security (biometric verification).

Machine vision is the industrial application of vision technology. Machine vision can be described as the understanding and interpretation of images obtained from a vision system or camera. The use of machine vision in industrial automation has increased over the years and now occurs in virtually all manufacturing industries.

Here are a few examples of machine vision applications:

- Guiding robots so that they can adapt to their environment
- Verification of welds
- Identification of people
- Inspection of circuit boards for correct electronic component placement
- Reading barcodes
- Ensuring a lid is placed on a spray bottle (part presences)
- Measurement of parts.

A machine vision system may consist of various hardware configurations suited to different applications. A vision system could consist of either one or many cameras. The additional use of a light source may be used to improve image quality. Cameras are available with various resolutions and optics best suited for specific applications. The precision of the inspection depends on the working distance, the field-of-view (FOV) and the number of physical pixels in the camera's sensor. The camera contains a highly specialized sensor that

is more expensive than a web cam. The cameras can be triggered by the machine vision system to obtain an image by a part-in-place sensor. Furthermore, the cameras have sophisticated exposure and fast electronic shutters that can freeze the motion of most parts. The selection of the correct camera and lens plays an important role when designing a machine vision system.

The use of a vision system in a RAS may not only be to identify parts and locate part coordinates for a robot, but also for quality control. The majority of manufactures use quality control inspections to reduce the production of defective parts. These quality control measures can be conducted in line at different points in the production process or just for a final check of the final product. The use of multiple quality checks helps identify defective parts quickly, earlier on in the product production. This saves time and manufacturing resources by removing these products from the production line. Quality control practises help reduce the number of defective products sold to the consumer.

Vision systems are easily reconfigurable from one application to the next. Vision systems have the flexibility to switch between different inspection programs depending on the task required. They are far quicker and more accurate at performing inspections than humans.

One can, therefore, conclude that a RAS should incorporate a vision system as they are easily reconfigurable from one application to the next and able to perform multiple operations if required. They require minimal hardware reconfiguration unless the camera position and/or lens need to be changed to meet the requirements of a particular application. Reconfiguration of a vision system lies mainly at a software level, when inspections need to be added or modified. They are initially expensive to implement, but require minimal running costs.

2.5. Communication Hardware

The selection of a common or generic communication medium plays an important role in a RAS as it will allow multiple devices from different vendors to communicate with each other

without the need for additional hardware or adapters. A communication medium that meets this requirement is the Ethernet network. Ethernet is not only used in the connection of high-level networks, but is now becoming an industry standard in low-level networks.

Many PLC's produced today come standard with an Ethernet port that is not only used for programming, but also has the ability to connect to 3rd party applications via Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). This allows the PLC to be integrated directly into the high level control system allowing the high level control system to read/write data from/to the PLC.

2.6. Chapter References

[1] Koren, Y., 2006, *General RMS characteristics*, Comparison with dedicated and flexible systems, Chapter 3 in Reconfigurable Manufacturing Systems and Transformable Factories, Springer.

[2] Setchi, R. M. and Lagos, N., 2004, *Reconfigurability and Reconfigurable Manufacturing Systems – State-of-the-art Review*, 2nd IEEE International Conference of Industrial Informatics: Collaborative automation – one key for intelligent industrial environments, Berlin, June 2004.

[3] ELMaraghy, H. A., 2006, *Flexible and reconfigurable manufacturing systems paradigms*, International Journal of Flexible Manufacturing Systems, Vol 17, pages 261-276.

[4] Koren, Y., 1983*, Computer Control of Manufacturing Systems*, McGraw Hill, New York.

[5] Mehrabi, M. G., Ulsoy, A. G., Koren, Y., Heytler, P., 2002, *Trends and perspectives in flexible and reconfigurable manufacturing systems*, Journal of Intelligent Manufacturing, Vol 13, pages 135-146.

[6] [Online] Available from http://www-personal.umich.edu/~ykoren/uploads/Design_of_Reconfigurable_Manufacturing_Systems.pdf [Accessed 11 June 2012]

[7] Maier-Speredelozzi, V., Koren, Y., Hu S. J., 2003, *Convertibility measures for manufacturing systems*, CIRP Annals, Vol 51, no 1, pages 367-371.

[8] Yu, F., Yin, Y., Sheng, X., Chen, Z., 2003, *Modelling strategies for reconfigurable assembly systems*, Assembly Automation Vol 23, no 3, pages 266-272.

[9] Koren, Y., 2004*, Reconfigurable Manufacturing Systems*, COMA annals, Vol 1, pages 69-79.

[10] Molina, A., Rodriguez, C. A., Ahuett, H., Cortes, J. A., Ramirez, M., Jiminez, G., Martinez, S., 2005*, Next-generation manufacturing systems: key research issues in developing and integrating reconfigurable and intelligent machines*, International Journal of Computer integrated manufacturing, Vol 18, no 7, pages 525-536.

[11| Basson, A., Vermaak, H., Kallis, D., 2008, *Reconfigurable Assembly Systems: A Literature Review*, Final Milestone Report for AMTS project AMTS-07-10-P

[12] Hains, C. L., 1985, *An algorithm for carrier routing in a flexible material-handling system*, IBM Journal of Research and Development, Vol 29, no 4, pages 356-362.

[13] Le-Anh, T. and De Koster, M., 2006, *A review of design and control of automated guided vehicles systems*, European Journal of Operational Research, Volume 171 no 1, pages 1-23.

Chapter 3: Development and Implementation of a Reconfigurable Assembly System

3.1 Introduction

This section looks at the different components used in the development of the assembly system. It also focuses on how these components interface with each other and the techniques used to integrate these components.

3.2 Assembly System Overview

The assembly system was designed to have both reconfigurable hardware and software components that provide customized flexibility for a particular family of parts. Figure 3.1 shows the assembly system that was constructed for this research project.

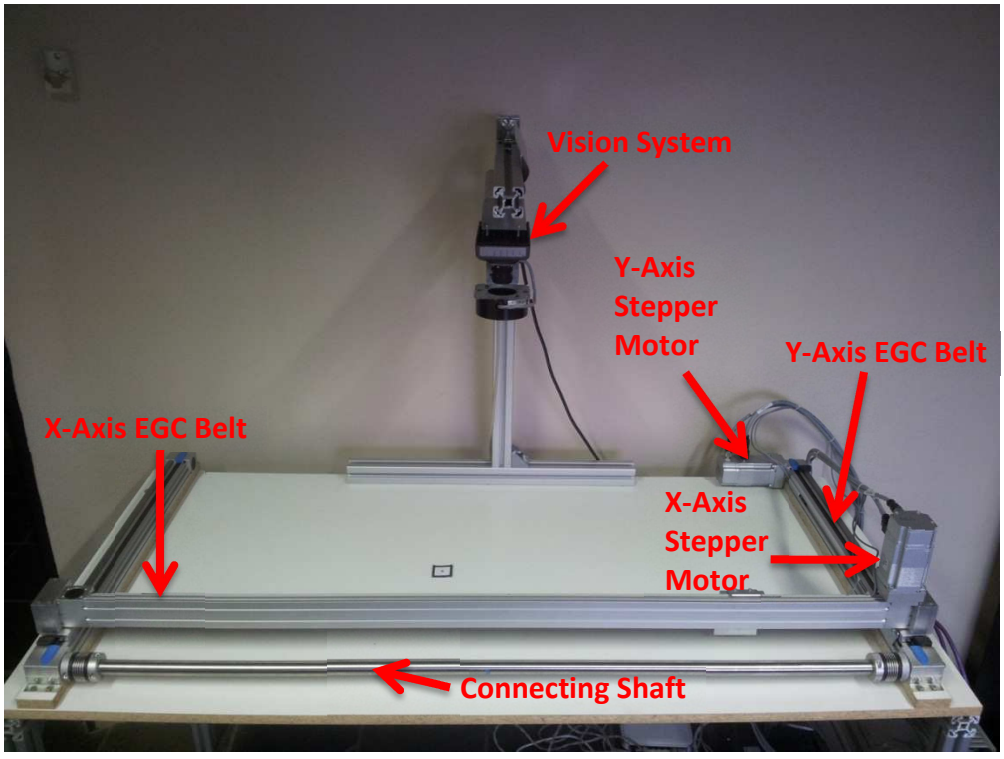


Figure 3.1: Assembly system

The Festo linear axes were used as it provides reconfigurability to the system. The Festo stepper motor with encoders creates a closed loop control system in conjunction with the Festo stepper motor controller. This gives the system a high degree of accuracy required for pick-and-place applications. Other factors, when considering the Festo axis, include the ease of use, integration, setup and familiarity with the Festo products.

The National Instruments NI 1742 Smart Camera was chosen, as it is able to hold many different inspections that can be selected depending on the product that needs to be built or the task that needs to be carried out. The vision system configuration software, Vision Builder for Automated Inspection (VBAI), allows the user to configure sophisticated inspections that include loops and branches without programming [2]. The smart camera also provides and Ethernet interface for communication to other control systems over various industrial protocols [2].

The PLC control system made use of a Mitsubishi Programmable Logic Controller (PLC). This entry level controller can communicate over multiple different industrial protocols that will allow the assembly system to communicate with many different devices from different vendors. This gives the system scalability and modularity. Since the Mitsubishi products and integration of the products are familiar to me, it was the perfect choice when deciding on a suitable controller. Figure 3.2 shows the control system panel and the different components used to control and communicate with the different devices used in the system.

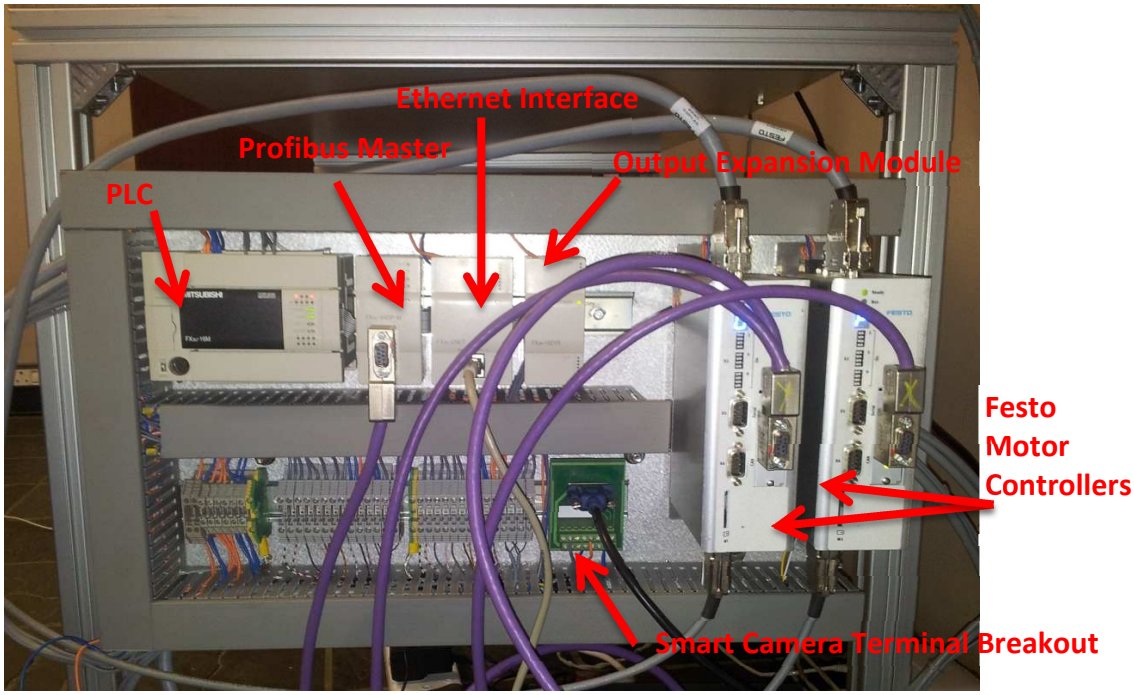


Figure 3.2: Control system panel and components

Figure 3.3 shows the network layout of the system indicating the devices and the communication mediums used.

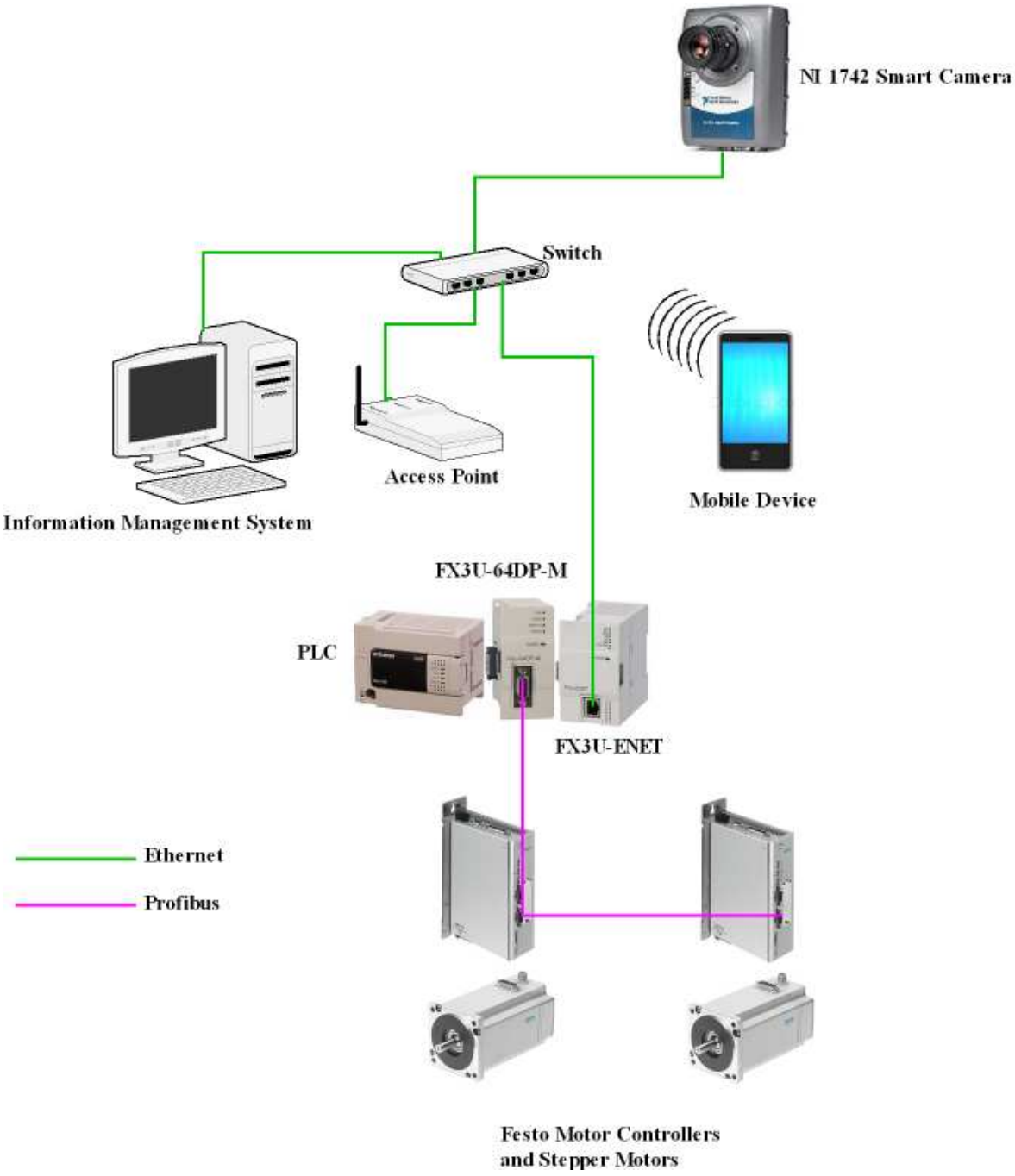


Figure 3.3: Network layout

3.2.1 Product Description

The assembly system is designed to assemble multiple product variations. The product consists of a base pallet that has twelve available positions. Figure 3.4 shows the base pallet and the twelve open positions.

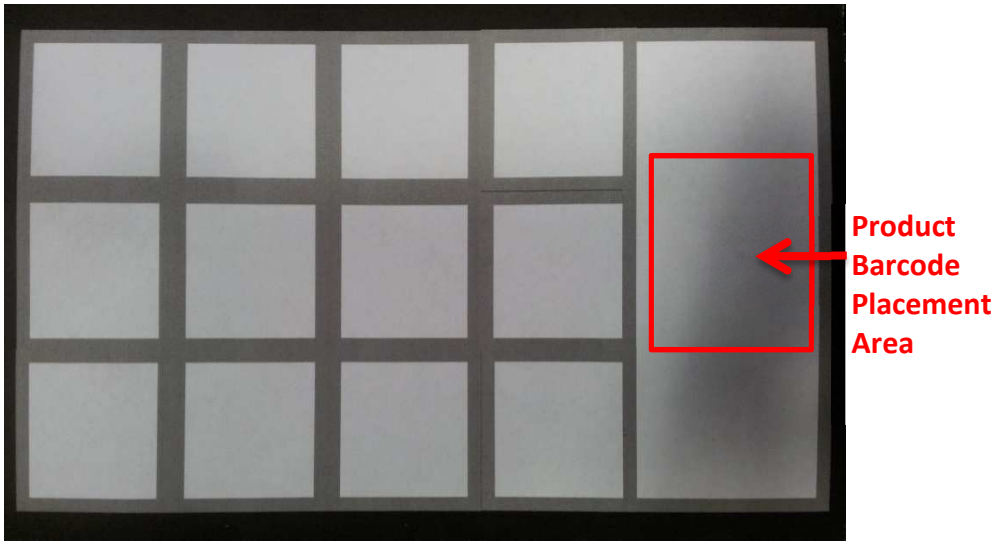


Figure 3.4: Base pallet

Each position on the base pallet may consist of three different parts or the position may remain empty depending on the product configuration that needs to be built. Figure 3.5 shows the three parts available for the product family.

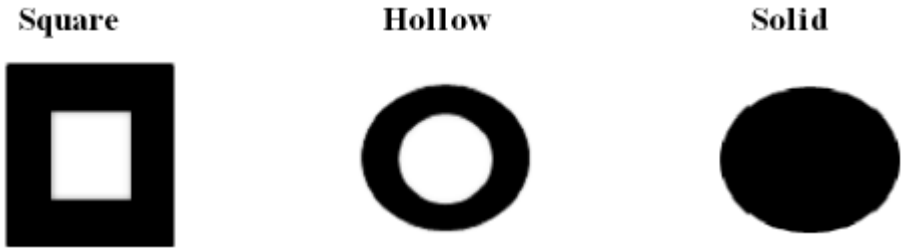


Figure 3.5: Types of product parts

3.2.2 X-Axis Hardware

The X-Axis is made up of one Festo EGC-50-1085-TB-KF-0H-GK toothed belt linear axis. The linear axis has a length of 1085 mm. This type of linear axis has a repetition accuracy of ± 0.008 mm [1]. The X-Axis is driven by a two-phase Festo EMMS-ST-57-S-SE stepper motor that has an integrated encoder for closed loop control. The stepper motor is controlled by using a Festo CMMS-ST-C8-7 motor controller. The motor controller can be operated via

digital and analogue control signals and networked via the integrated CAN bus. There are various other field bus systems available for the motor controller that makes it a perfect choice for a RAS. I chose to use a Profibus field bus system as I am familiar with the setup and control of Profibus devices, and it is also a network that is commonly used throughout industry applications.

The X-Axis also has two inductive proximity sensors located at each end of the linear axis that are used as travel limit sensors. These sensors prevent the X-Axis from over-travel and are also used to home the X-Axis before operation. The sensors are wired as normally open. Homing is used to reset the encoder value to zero. This ensures accurate position control of the axis.

The parameterisation of the X-Axis is done using the FCT (Festo Configuration Tool). This tool enables the configuration and commissioning of the CMMS-ST motor controllers using a RS 232 cable.

3.2.3 Y-Axis Hardware

The Y-Axis is made up of two EGC-50-480-TB-KF-0 H-GK toothed belt linear axes. Both linear axes are 480mm in length. The two EGC belts are placed in parallel and are coupled by a connecting shaft to synchronize the motion of the two axes. The X-Axis is mounted to the Y-Axis on either end. One of the axes is driven by a two-phase stepper motor that has an integrated encoder for closed loop control. The stepper motor is also controlled by a CMMS-ST motor controller. This motor controller also makes use of the optional Profibus field bus system for positioning.

One of the axes contains two inductive proximity sensors located at each end of the linear axis and are used as travel limit sensors and for homing. These sensors are also wired in a normally open configuration. Parameterization of the Y-Axis makes use of the FCT software.

3.2.4 Vision System

The assembly system makes use of a National Instruments NI1742 Smart Camera. The smart camera is used to scan barcodes, obtain position coordinates and verify that the product as assembled is correct. The Data Matrix barcode located on the base pallet contains the product ID and a unique build ID. This information is transmitted to the Information Management System to obtain the task that must be completed for the specific product and also for product traceability. The position coordinates are used by the X-Axis and Y-Axis for the correct placement position for the parts.

The smart camera uses VBAI (Vision Builder for Automated Inspection) installed on a PC to program the individual inspections. The smart camera can contain many different inspections located on its own internal hard drive. This enables multiple inspections to be loaded for many different products. The NI1742 smart camera supports both Ethernet and serial based industrial protocols such as Modbus TCP [2]. The camera allows the user to select the desired inspection via one of the following methods:

- Digital Inputs
- Serial port
- Ethernet

The inspection selection will take place over Ethernet for this study as the Information Management System is already obtaining data from the inspections over Ethernet and it is not necessary to implement another communication protocol.

3.2.5 PLC Control System

The control system makes use of a Mitsubishi FX3U PLC (Programmable Logic Controller) which is widely used in applications ranging from machine control to networked systems [4]. The FX3U PLC has the ability to be expanded to adapt to the changing needs of the required application.

The control system consists of the following Mitsubishi hardware components:

- PLC (FX3U-16M)
- Profibus Master (FX3U-64DP-M)
- Ethernet Module (FX3U-ENET)
- Output Module (FX2N-16EYR).

A Mitsubishi FX3U PLC (Programmable Logic Controller) is used to control and monitor the X-Axis and Y-Axis CMMS-ST stepper motor controllers over a Profibus network. It is also used to read digital feedback signals from the motor controllers and switch on digital control signals that enable the motor controllers for operation. The smart camera receives digital input from the PLC to acquire an image and perform an inspection. One of the PLC's outputs is connected to the TrigIn+ (Pin Number 2) terminal of the smart cameras terminal breakout board input terminals for triggering the selected inspection.

The PLC is programmed in Ladder Logic using Mitsubishi GX Developer. The PLC has an Ethernet expansion module connected that allows the PLC to communicate to other systems via TCP/IP or UDP. The Ethernet module includes the ability to have eight simultaneous open connections [4]. This module is used to integrate the system into the Information Management System and control certain functions (e.g. start, stop, reset, manual control, etc.) as well as provide the same functionalities to mobile devices that are running the application.

3.3 PLC and Festo Motor Controller Interfacing

3.3.1 Festo Stepper Motor Controllers

Festo has developed an optimised data profile especially tailored to the target applications of handling and positioning tasks, the "Festo Handling and Positioning Profile (FHPP)" [3]. The PLC will make use of the FHPP over Profibus and pass control bits, position coordinates, speed set point and read the motor controller's status bits. In order to support the FHPP over Profibus on the Festo motor controller, the following items were incorporated from

Festo and Mitsubishi: A Festo CAMC-PB Profibus interface module with a 9-pin DSUB socket and a Mitsubishi FX3U-64DP-M Profibus master expansion card with a 9-pin DSUB socket. The FX3U-64DP-M Profibus card is used as the master and the Festo CAMC-PB cards are used as slaves.

3.3.2 Festo Handling and Positioning Profile (FHPP)

The FHPP can either be operated in record select mode or direct mode. Record select mode makes use of user defined traversing records that are saved in the controller. Direct mode is more flexible and allows position, speed and torque set points to be transferred directly in the I/O telegram over Profibus. This makes it possible to adjust to system changes in tool sizes and product variation without having to change records in the motor controller. Direct mode will be used in this study, as it allows greater flexibility for any system changes and product variation.

The FHPP works by exchanging data (8 bytes) of control and status information with the PLC. The data structure of the FHPP control and status bytes for direct mode is indicated in Table 1.

Table 1: Direct mode FHPP I/O data

Direct Mode								
	Byte 1	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**
Output Data	**CCON**	**CPOS**	**CDIR**	**Nominal Value 1**	**Nominal Value 2**			
Input Data	**SCON**	**SPOS**	**SDIR**	**Actual Value 1**	**Actual Value 2**			

The output data are the control bytes and these control the motor controllers' operation. The PLC is responsible for transferring the output data to the motor controller via Profibus. The input data are the status bytes transferred from the motor controller to the PLC via Profibus. A breakdown of the individual bits of these bytes is indicated in Table 2 and Table 3.

Table 2: FHPP Control Bytes Overview

Control Bytes								
CCON	**B7** **OPM2**	**B6** **OPM 1**	**B5** **LOCK**	**B4**	**B3** **RESET**	**B2** **BRAKE**	**B1** **STOP**	**B0** **ENABLE**
	FHPP operation mode selection		Software access blocked	Not Used	Acknowledge fault	Release brake	Stop	Enable drive
CPOS	**B7**	**B6** **CLEAR**	**B5** **TEACH**	**B4** **JOGN**	**B3** **JOGP**	**B2** **HOM**	**B1** **START**	**B0** **HALT**
	Not Used	Delete remaining path	Teach value	Jog negative	Jog positive	Start homing	Start positioning job	Halt
CDIR	**B7**	**B6**	**B5** **XLIM**	**B4** **VLIM**	**B3** **CONT**	**B2** **COM2**	**B1** **COM1**	**B0** **ABS**
	Not Used	Not Used	Deactivate stroke limit	Velocity limit reached	Continuous tracking mode	Control mode (position, torque, speed)		absolute/ relative

Table 3: FHPP Status Bytes Overview

Status Bytes								
SCON	**B7** **OPM2**	**B6** **OPM 1**	**B5** **LOCK**	**B4** **24VL**	**B3** **FAULT**	**B2** **WARN**	**B1** **OPEN**	**B0** **ENABLED**
	FHPP operation mode acknowledge		Device control software	Load voltage applied	Fault	Warning	Operation enabled	Drive enabled
SPOS	**B7** **REF**	**B6** **STILL**	**B5** **DEV**	**B4** **MOV**	**B3** **TEACH**	**B2** **MC**	**B1** **ACK**	**B0** **HALT**
	Drive referenced	Downtime monitoring	Drag error	The axis is moving	Teach acknowledge ment	Motion Complete	Start acknowledg ment	Halt
SDIR	**B7**	**B6**	**B5** **XLIM**	**B4** **VLIM**	**B3** **CONT**	**B2** **COM2**	**B1** **COM1**	**B0** **ABS**
	Not Used	Not Used	Stoke limit reached	Velocity limit reached	Continuous tracking mode	Control mode acknowledgment (position, torque, speed)		absolute/ relative

3.3.3 CMMS-ST Motor Controller Profibus Configuration

Each motor controller was installed with the Festo CAMC-PB Profibus option card. The Profibus address for each motor controller node was assigned by setting the dipswitches located on the front of the motor controller. The Profibus address assigned to the Y-Axis and the X-Axis are three and four respectively.

Each motor controller was set to use the Profibus DP control interface using FCT (Festo Configuration Tool). The baud rate does not need to be set as the Festo Profibus option card is able to detect the baud rate of the master Profibus device automatically.

3.3.4 FX3U-64DP-M Profibus Master Configuration

The PLC has a FX3U-64DP-M Profibus master expansion card connected to it. This enables the PLC to communicate with other Profibus slave devices located on the same network. The motor controllers require Profibus master systems to use the FHPP for communication. This protocol is contained in a GSD file specific to the Festo motor controller and can be downloaded from the Festo website.

The Profibus master was configured using Mitsubishi GX Configurator DP. This software is used to add Profibus slave devices, set Profibus addresses, set communication parameters and also monitor the Profibus slave device that are currently connected to the Profibus master expansion card. The GSD file for the FHPP contains two profiles, namely “FHPP Standard” and “FHPP Standard + FPC”. The latter allows the PLC to change some of the motor controller parameters that are normally set up using the FCT. The “FHPP Standard” profile will be used for this study.

The Profibus master is configured to use 1.5 Mbits/s baud rate. This baud rate is sufficient for this application’s data requirement. Once the Festo motor controller GSD file was added to the GX Configurator DP software, the X-Axis was added as a slave with Profibus address four and set to use the “FHPP Standard” profile. The Y-Axis was set to use Profibus address three and it was also set to use the “FHPP Standard” profile. This Profibus configuration was downloaded to the Profibus Master via the PLC using the Mitsubishi SC-09 programming

cable connected to the PLC programming port. There is some additional PLC code required to start the transfer of data between the master and the slave devices. This code not only starts the data transfer, but is also used to assign PLC data memory areas to the slave devices for read/write operations. Figure 3.6 shows an example of the PLC ladder code that moves the data in memory area D1200 – D1203 (4 words) to the first addressable slave device memory area (U0\G2350) on the Profibus network.

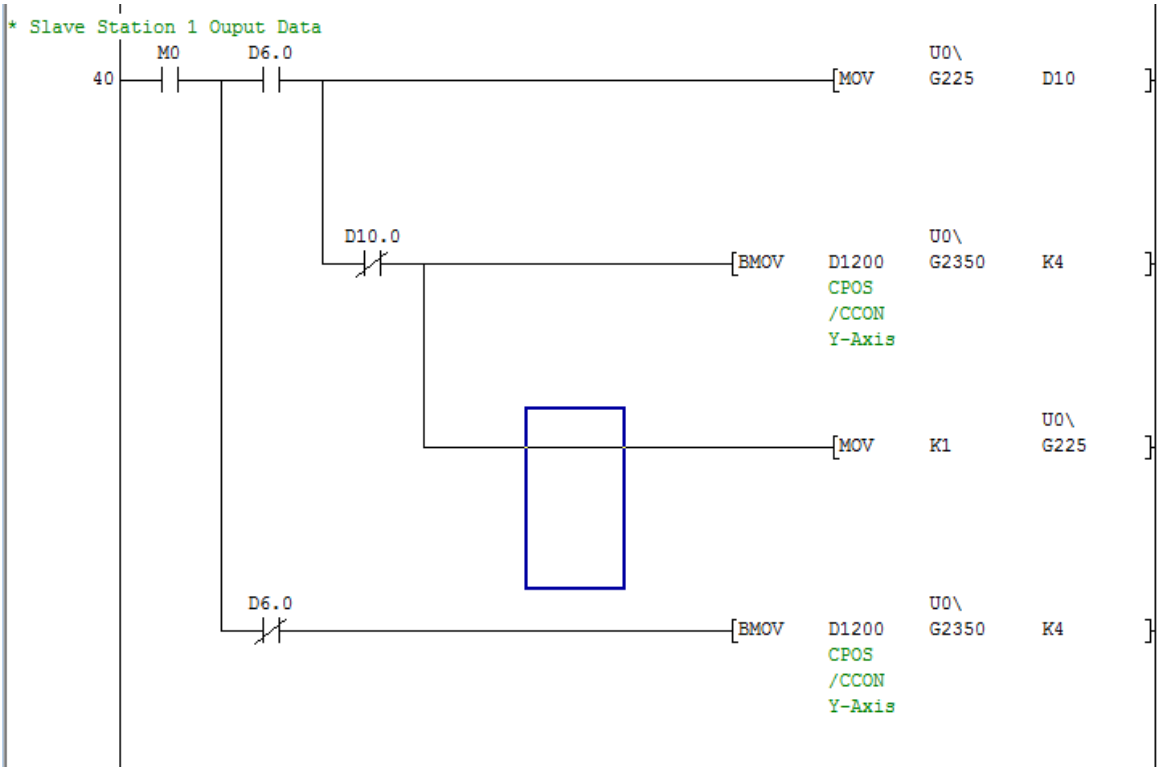


Figure 3.6: Ladder example of sending data to slave device

3.4 National Instruments Smart Camera

3.4.1 Introduction

The smart camera control is split up into three parts: the inspections loaded on the smart camera, the inspection selection and the smart camera agent embedded in the Information Management System that communicates with the smart camera.

The National Instruments NI1742 Smart Camera has been set up using the VBAI (Vision Builder for Automated Inspection) to act as a TCP Server. The TCP Server will listen on port 4210 for any connection requests from a TCP Client namely the Smart Camera Agent. Data that is sent from the smart camera to the Information Management System is transmitted over this port. This port is also used to transmit data from the Information Management Systems to the smart camera to select different inspections depending on the required task.

3.4.2 Inspections

There are five different inspections loaded into the smart camera. The Find Task inspection is the default inspection loaded when the smart camera is powered up. The smart camera continually runs a background program that checks for inspection triggers and also looks for any requests from the Information Management System to change to a different inspection. The flow diagram for the background program is shown in Figure 3.7. The following inspections are loaded on the smart camera:

- Find Task – Look for a valid Data Matrix barcode and transmit the barcode data to the Information Management System to determine the task to be carried out.
- Check Empty – Check all twelve positions on the base pallet to determine if they are empty and transmit the result back to the Information Management System.
- Find Positions – Finds the centre point of each one of the twelve positions and transmits the x, y and angle of the position to the Information Management System.
- Verify – Transmits the type of part found in each position back to the Information Management System for verification.
- Calibration – The calibration inspection is used to find the x and y real world coordinates of a known object to calibrate the X-Axis and Y-Axis coordinates with the smart camera's coordinates. This is an important inspection that helps with positioning accuracy. The x and y coordinates are transmitted to the Information Management System and the saved in a SQL database to be used as a calibration reference for the X-Axis and Y-Axis for positioning tasks.

All the above inspections are discussed in detail with flow charts in the sections to follow.

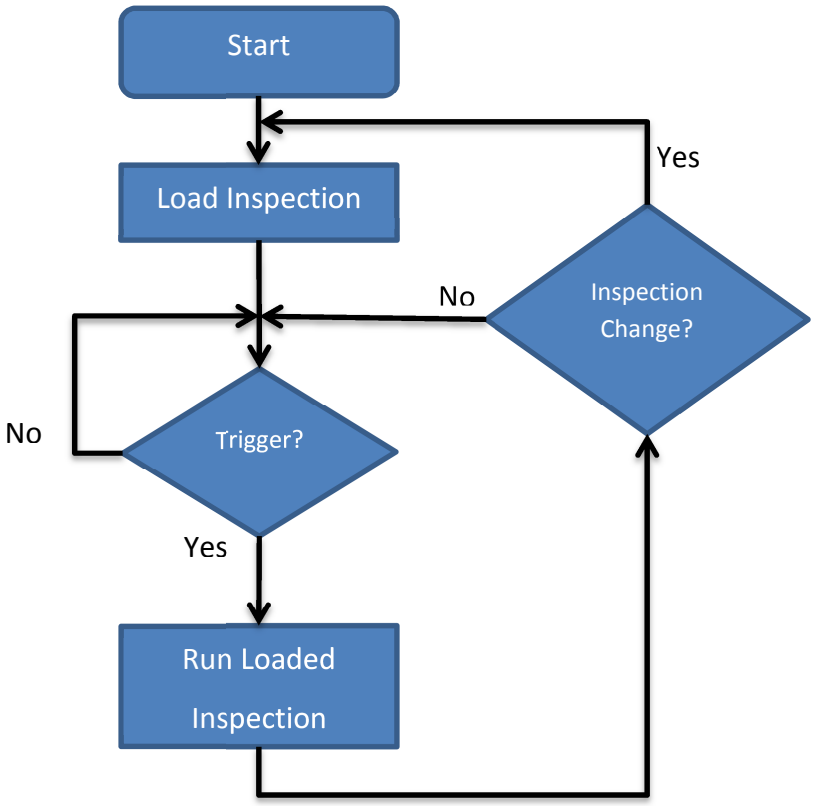


Figure 3.7: Flow diagram of the smart camera background program

3.4.2.1 Find Task Inspection

The Find Task Inspection is used to scan the Data Matrix barcode located on the base pallet. This Data Matrix barcode contains the product ID and build ID information. The barcode data structure of the embedded code is an alphanumeric string consisting of fourteen characters. An example of a Data Matrix code with string “#1234123A567BC” embedded is shown in Figure 3.8.

Data Matrix codes were chosen as they offer wide reading angles and high reliability due to error correction algorithms. Up to 30% of the code can be damaged while it would still be possible to read the code. Data Matrix codes are also able to stored more bytes than traditional 1D barcodes. [5]

Figure 3.8: Data Matrix Barcode

In order for the correct product and build to be identified in the RAS, it was necessary to create a predefined barcode structure that the smart camera, Information management System and operator are able to interpret. The following describes the barcode structure in detail:

- The "#" prefix is used to identify that the barcode being scanned is a valid barcode for the system and not some random barcode placed on the product.
- The product ID consists of four characters unique to that product type.
- The build ID consists of nine characters, is unique to that product build and used for traceability purposes.

A breakdown of the barcode data is shown in Table 4.

Table 4: Barcode Data Structure

Barcode Data Structure Example		
Position	**String**	**Description**
1	#	All barcodes must be prefixed with this character
2	1	Product ID (Alphanumeric)
3	2	Product ID (Alphanumeric)
4	3	Product ID (Alphanumeric)
5	4	Product ID (Alphanumeric)
6	1	Build ID (Alphanumeric)
7	2	Build ID (Alphanumeric)
8	3	Build ID (Alphanumeric)
9	A	Build ID (Alphanumeric)
10	5	Build ID (Alphanumeric)
11	6	Build ID (Alphanumeric)
12	7	Build ID (Alphanumeric)
13	B	Build ID (Alphanumeric)
14	C	Build ID (Alphanumeric)

The smart camera, as discussed in section 3.4.2, by default will start up with the Find Task inspection whenever it is powered on. The inspection will wait for an external trigger from the PLC. The trigger will result in the smart camera acquiring an image and then performing the inspection to located barcodes that are prefixed with "#". The flow diagram is presented in Figure 3.9.

In the event of a trigger from the PLC, the smart camera program starts out by initializing all smart camera variables (gain, exposure time, lighting and trigger). These variables have been set up in the smart camera using the VBAI software. The next step for the smart camera is to acquire the image for analysis. The smart camera then applies some image processing algorithms to the image that makes it easier to look for the left and bottom edge of the base pallet. The intersection of these two edges defines the coordinate system for the region of interest where the smart camera should search for a barcode prefixed with "#".

If no barcode was found containing the "#" prefix then a string with the message "*STX;FindTask;Failed;ETX*" will be transmitted over the TCP/IP socket to the Information Management System. In the event of a barcode with prefix "#" located in the defined region of interest, then a string containing the message "*STX,FindTask;BarcodeData;ETX*" will be transmitted to the Information Management System. The Information Management System will then query a SQL database to determine the next step in the process depending on the barcode data retrieved and the current abilities of the assembly system that requested the task. A breakdown of the string transmitted to the Information Management System from the smart camera can be seen in Table 5.

Table 5 Find Task Message Structure

Find Task Message	
Position 1	STX (Start of Text)(Hex 0x02)
Position 2	Name of the Inspection
Position 3	Depending on the result of the inspection • Failed Inspection (*Failed*) • Successful Inspection (e.g. *#1234123A567BC*)
Position 4	ETX (End of Text)(Hex 0x03)

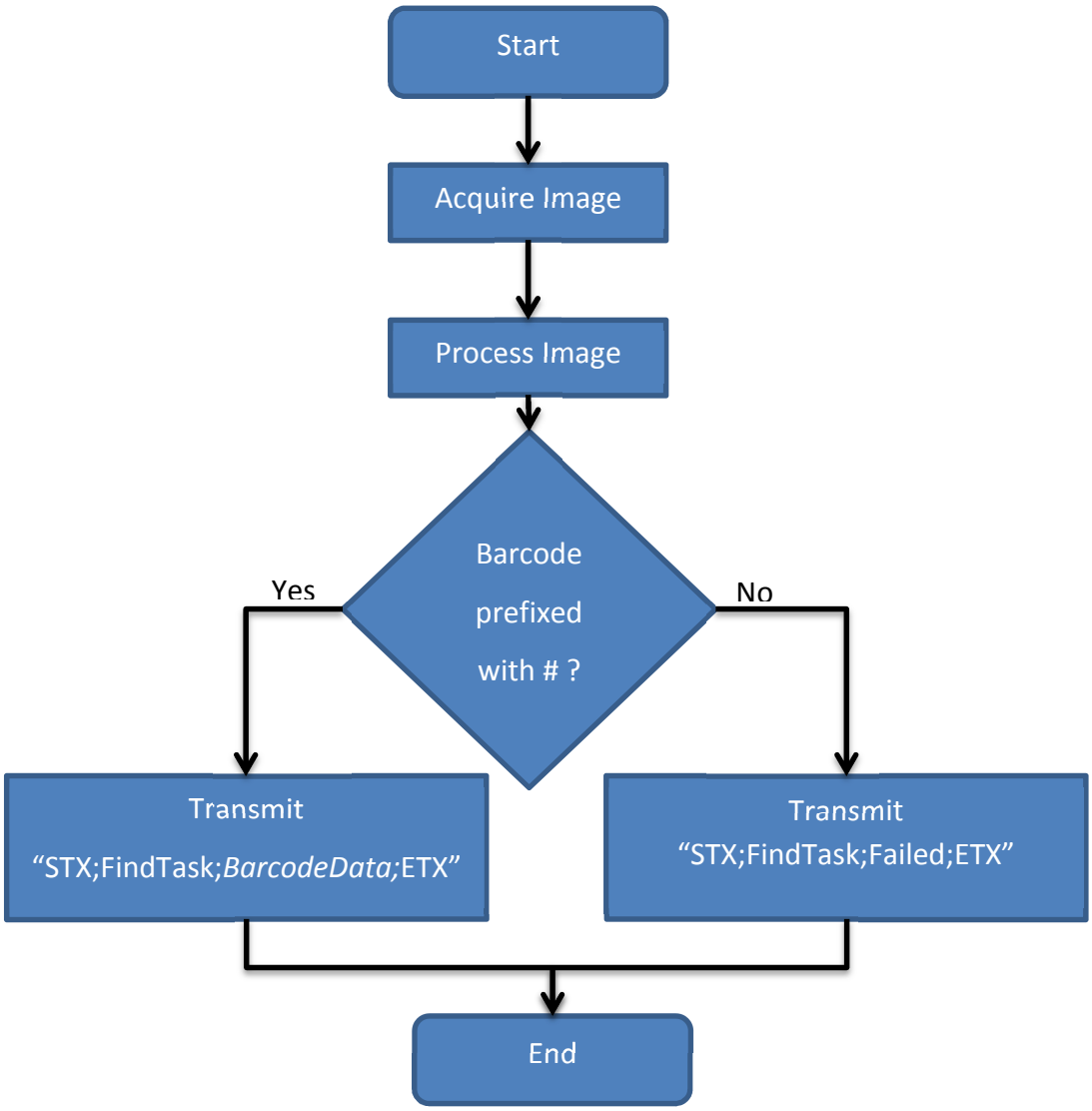


Figure 3.9: Flow diagram of the Find Task inspection

3.4.2.2 Check Empty Inspection

The Check Empty Inspection is used to check that all twelve positions of the base pallet are empty before continuing with the build required product task. This inspection is loaded after the Information Management System confirms that the barcode is a valid barcode and the task required for the specific barcode is either that the product be i) built or ii) built and verified.

The Check Empty inspection will run once the external trigger has been activated by the PLC. The flow diagram for this inspection is presented in Figure 3.10. In the event of a trigger

from the PLC, the smart camera program starts out by initializing all smart camera variables (gain, exposure time, lighting and trigger). These variables have been set up in the smart camera using the VBAI software. The next step for the smart camera is to acquire the image for analysis. The smart camera then applies some image processing algorithms to the image that makes it easier to look for the left and bottom edge of the base pallet.

The VBAI program will then compute the intersection coordinates of these bottom and left edges. The intersection coordinates will be used as the reference coordinate system for all steps in this inspection. The coordinate system will allow the region of interest defined in the program to move dynamically in respect of the reference coordinate system. This means that the pallet does not have to be placed in a fixed position, but can be moved around within the camera's field of view. The first position of the base pallet is checked by measuring the intensity of the pixels for that position. If the average intensity value is between 245.00 and 255.00 then the position is determined to be empty and the empty counter is incremented by one. The program will increment through all four columns of the first row of the base pallet before moving onto row two and then onto row three. Each time an empty position is detected, the empty counter is incremented by one. In the event that a position is determined to contain an object, then the empty counter will not be incremented and the next position will then be inspected.

After all the positions have been inspected, the program will check to see if the empty counter is equal to twelve. If the counter is equal to twelve, then it is determined that the base pallet is empty and the command "*STX;CheckEmpty;Empty;ETX*" will be transmitted to the Information Management System. This confirms that the base pallet is empty and the process can continue to the next step of building the product. The Find Positions inspection will be loaded at this point to retrieve all twelve position coordinates of the base pallet. If the counter is less than twelve, then the command "*STX;CheckEmpty;Failed;ETX*" will be transmitted to the Information Management System.

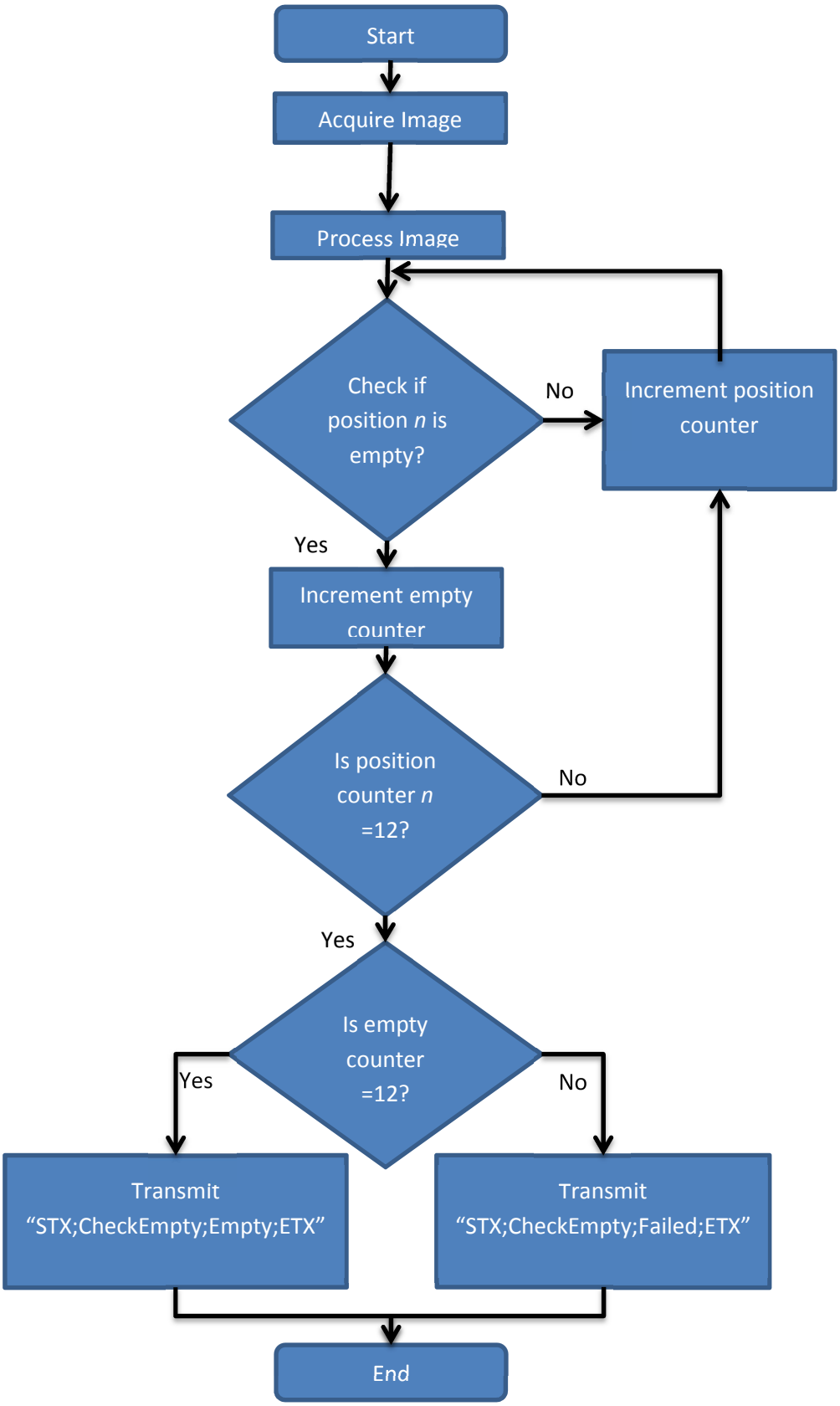


Figure 3.10: Flow diagram of Check Empty inspection

3.4.2.3 Find Positions Inspection

The Find Positions Inspection is used to retrieve the position coordinates of all twelve positions of the base pallet for positioning tasks. The coordinates retrieved from the smart camera consist of the x coordinate, y coordinate and angle. This inspection is loaded after the Information Management System confirms that the Check Empty inspection confirmed that the base pallet was empty and the task to be carried out requires that the product be built or built and verified.

The Find Empty inspection will run once the external trigger has been activated by the PLC. The flow diagram for this inspection is presented in Figure 3.14. In the event of a trigger from the PLC, the smart camera program starts out by initializing all smart camera variables (gain, exposure time, lighting and trigger). These variables have been setup in the smart camera using the VBAI software. The next step for the smart camera is to acquire the image for analysis. The smart camera then applies some image processing algorithms to the image that makes it easier to look for the left and bottom edge of the base pallet.

The program will then compute the intersection coordinates of these bottom and left edges. The intersection coordinates will be used as the reference coordinate system for all steps in this inspection. The coordinate system will allow the region of interest defined in the program to dynamically move in respect of the reference coordinate system. This means that the pallet does not have to be placed in a fixed position, but can be moved around within the camera's field of view. The inspection's geometric matching tool has been taught a reference image of what an empty position looks like. The geometric matching tools limit has been set to look for twelve matches. Figure 3.11 shows an example of the results screen seen in VBAI when testing the inspection to find the position coordinates. Figure 3.12 shows the coordinates that will be used as part of the string that is transmitted to the Information Management System.

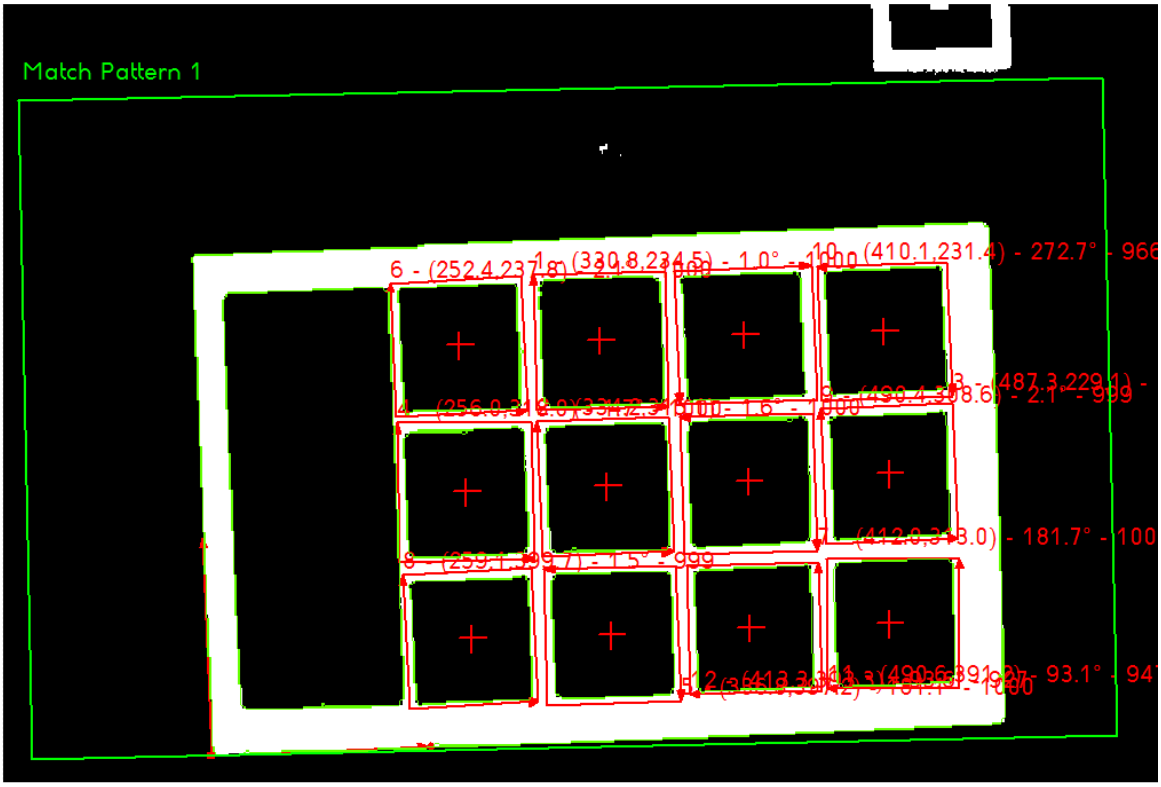


Figure 3.11: Find positions inspection example results

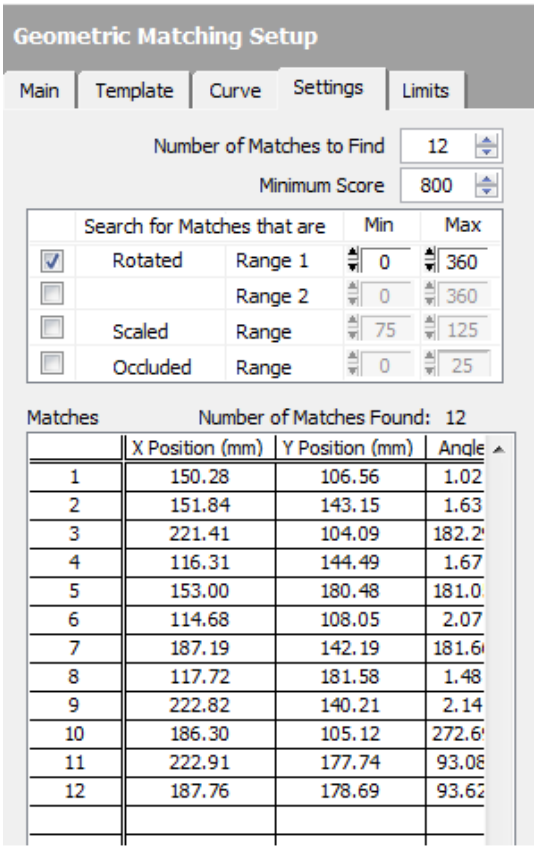


Figure 3.12: Find positions inspection coordinates

If the inspection locates twelve matches, then the position coordinates and angle of every position will be transmitted to the Information Management System. If the specific product being built requires verification and the assembly cell has the ability to verify the build, then the Verify Inspection will be loaded. Figure 3.13 represents the string transmitted after all twelve positions have been found. If the inspection locates fewer than twelve or even more than twelve matches, then the string that is transmitted to the Information Management System will be "*STX;FindPositions;Failed;ETX*".

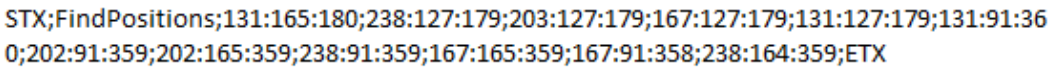


Figure 3.13: Find Position inspection string sample

The string transmitted when twelve matches are found is not in order from position one through to position twelve, so one should not assume that the first coordinates in the string represent position one. The smart camera places these variables in the order in which it was found. As a result of the string format, the Information Management System will break up the string and then sort them according to the x and y coordinates to determine which coordinates correspond to which position of the base pallet. Table 6 represent the string structure using the example string of Figure 3.13 when twelve positions have been located in the inspection.

Table 6: Find Positions inspection message structure

Find Positions Inspection Message		
Example String	*"STX;FindPositions;131:165:180;238:127:179;203:127:179;167:127:179;131:127:179;131:91:360;202:91:359;202:165:359;238:91:359;167:165:359;167:91:358;238:164:359;ETX"*	
Array Position	**String strip into an array**	**Description**
String Array[0]	STX	Start of Text
String Array[1]	FindPositions	Inspection Name
String Array[2]	131:165:180	x, y and angle
String Array[3]	238:127:179	x, y and angle
String Array[4]	203:127:179	x, y and angle
String Array[5]	167:127:179	x, y and angle
String Array[6]	131:127:179	x, y and angle
String Array[7]	131:91:360	x, y and angle
String Array[8]	202:91:359	x, y and angle
String Array[9]	202:165:359	x, y and angle
String Array[10]	238:91:359	x, y and angle
String Array[11]	167:165:359	x, y and angle
String Array[12]	167:91:358	x, y and angle
String Array[13]	238:164:359	x, y and angle
String Array[14]	ETX	End of Text

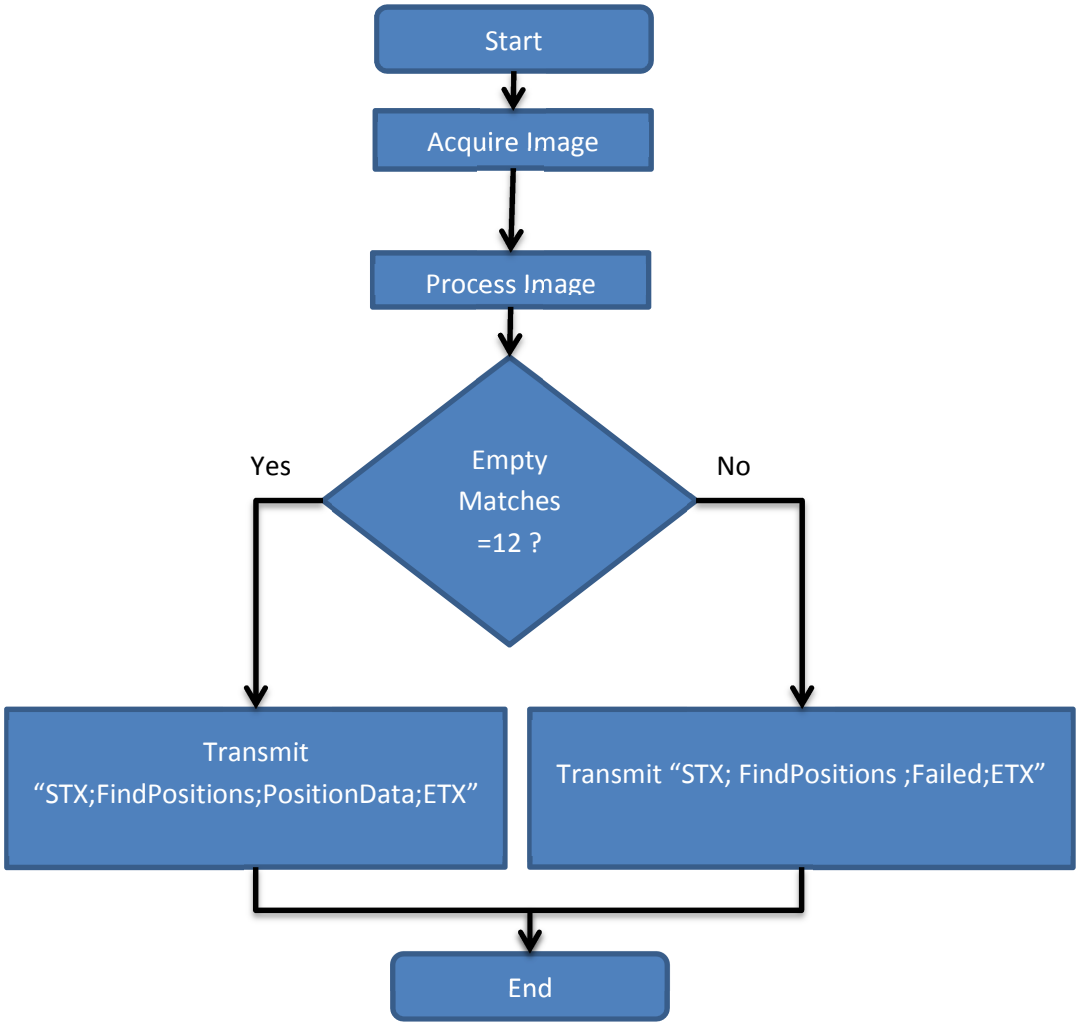


Figure 3.14: Flow diagram of Find Positions inspection

3.4.2.4 Verify Inspection

The Verify Inspection is used to retrieve the part type that is in each position of the base pallet in order for the Information Management System to compare the current build configuration with the configuration in the database for that specific product. This inspection can either be loaded after the smart camera found all twelve empty position coordinates as discussed in section 3.4.2.3 or if the task required for the base pallet is to only verify that the build configuration corresponds to the database build configuration.

The Verify inspection will run once the external trigger has been activated by the PLC. A trigger from the PLC will result in the smart camera initializing all variables related to exposure time, lighting, etc. All these variables have been previously set up using the VBAI software. The smart camera will then acquire an image for analysis. Image processing algorithms are then applied to the acquired image in order to improve the quality of edges and regions of interest required for this inspection.

The smart camera will locate the bottom edge and the left edge of the base pallet and then calculate the coordinates of the intersection of the two edges. The intersection coordinates will be used for all other steps in this inspection, as a reference coordinate, in order for the program to dynamically move regions of interest in respect to the location of the base pallet. The smart camera Verify inspection program will now locate position one of the base pallet and then use a match pattern algorithm to detect if the position contains a Solid, Hollow or Square part. The match pattern is taught to distinguish between the different parts. The position will also be checked to see if it is empty by measuring the intensity of pixels in the position. If the average intensity value is between 245.00 and 255.00 then the position is determined to be empty. Before a message is sent to the Information Management System, all the match patterns and empty results are sent to another step where it will perform some calculations. The logic used to determine if the part located in the position was a Solid, Hollow, Square, Empty or a foreign object can be seen in Figure 3.15. The foreign object variable is set if the program does not detect any of the three parts and if the measure intensity step does not find an empty position. Once the calculation has been completed, the smart camera will transmit a message string to the Information Management System regarding the results (e.g. *"STX;Verify;1;1;0;0;0;0;ETX"*). Table 7 shows a breakdown of the string structure. The program will then continue and check the next position until it has checked all twelve positions and then transmit *"STX;Verify;Finished;ETX"* which indicates that the inspection is complete. The flow diagram representing the Verify Inspection can be seen in Figure 3.16.

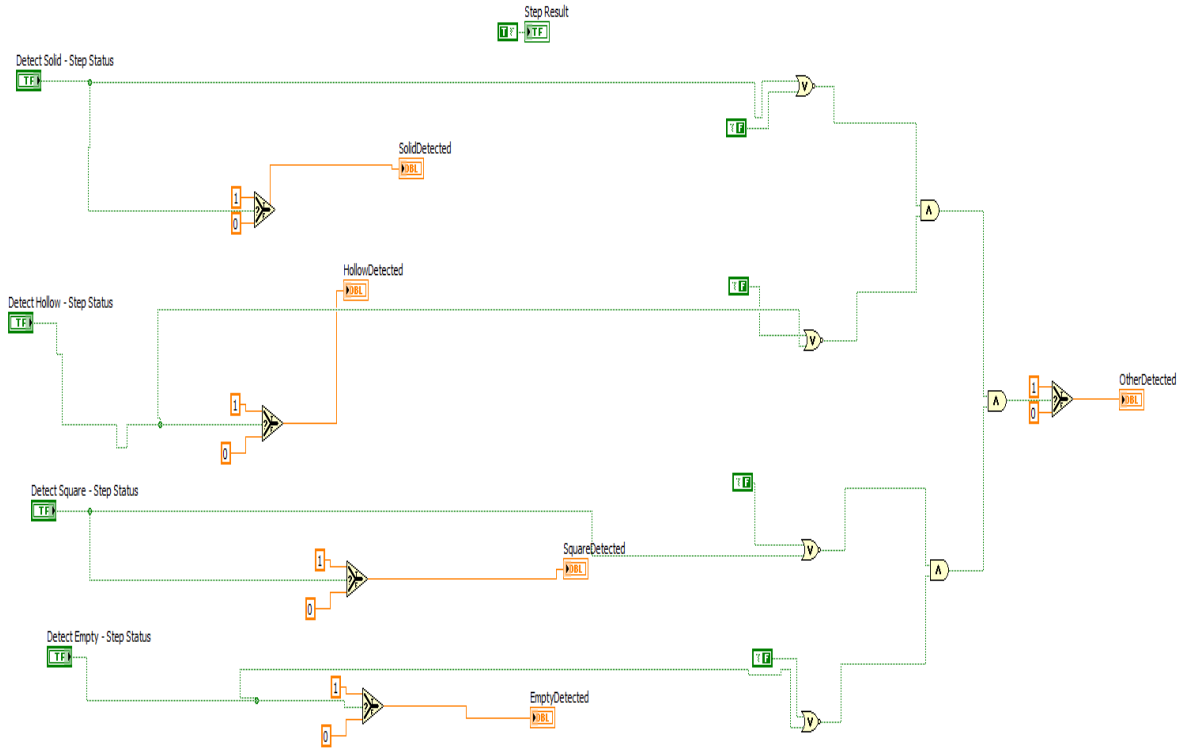


Figure 3.15: Verify inspection part detection calculation

Table 7: Verify Task message string structure

Verify Inspection Message		
Example String	*"STX;Verify;1;1;0;0;0;0;ETX"*	
Array Position	**String strip into an array**	**Description**
String Array[0]	STX	Start of Text
String Array[1]	Verify	Inspection Name
String Array[2]	1	Base Pallet Position Number
String Array[3]	1	Solid Part Detected
String Array[4]	0	Hollow Part Detected
String Array[5]	0	Square Part Detected
String Array[6]	0	No Part Detected (Empty)
String Array[7]	0	Foreign Part Detected
String Array[8]	ETX	End of Text

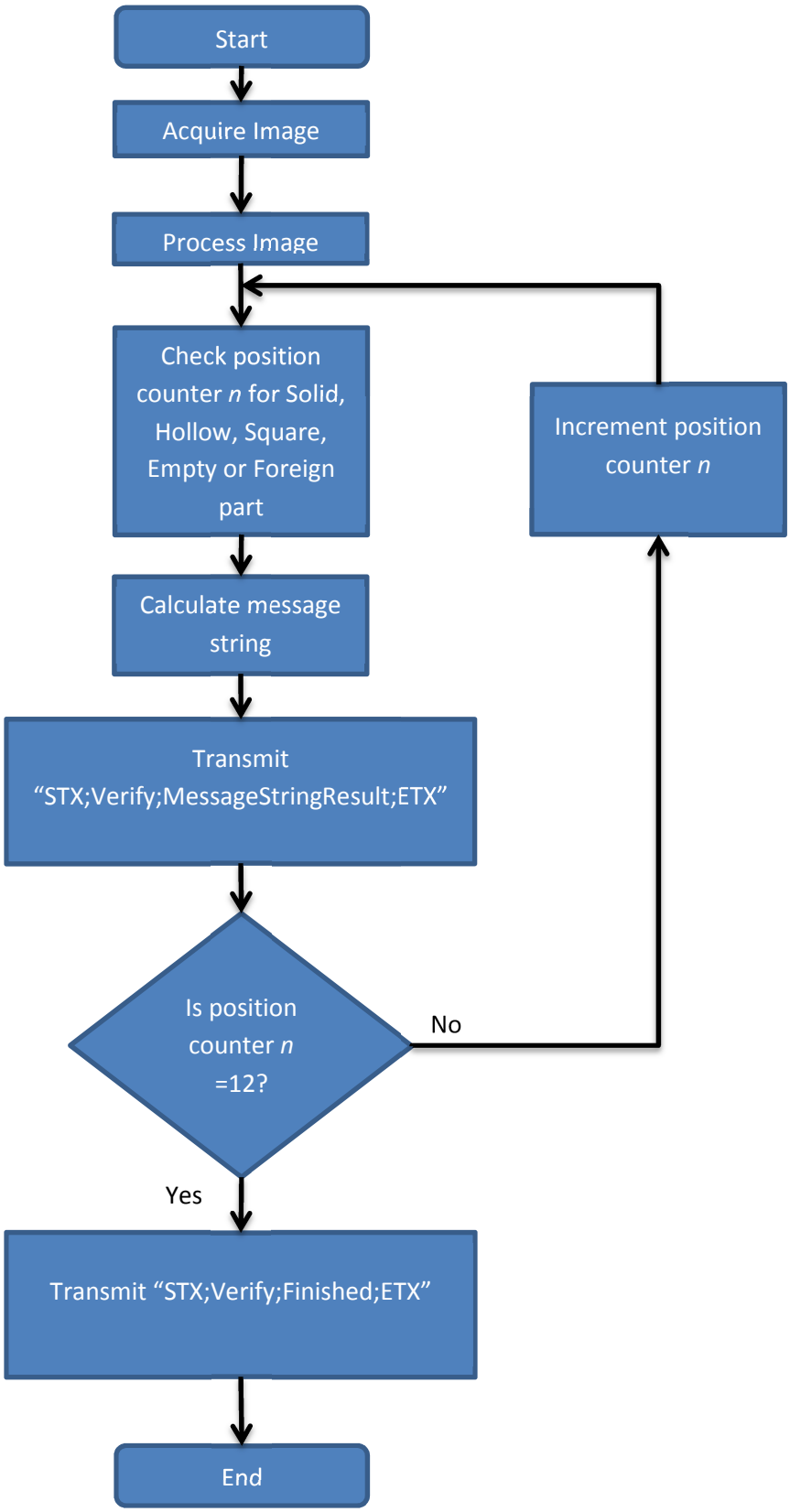


Figure 3.16: Flow diagram of Verify inspection

3.4.2.5 Calibration Inspection

The Calibration Inspection is used to calibrate the X and Y axis coordinates with the smart camera's pixel coordinates. This enables the X and Y axis to move to the correct position relative to the position of the camera when performing positioning tasks that require accuracy. This inspection is selected by an operator using the Information Management System.

The Calibration inspection will run once the external trigger has been activated by the PLC. A trigger from the PLC will result in the smart camera initializing all variables related to exposure time, lighting, etc. The smart camera will then acquire an image for analysis. Image processing algorithms are then applied to the acquired image in order to improve the quality of edges and regions of interest required for this inspection. The inspection will attempt to locate the bottom left coordinates of the calibration part, as seen in Figure 3.17.

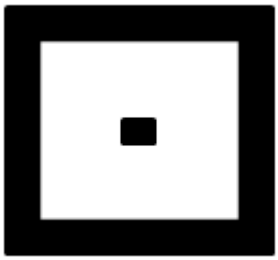


Figure 3.17: Part used for calibration

In order for the inspection to calculate the bottom left coordinates of the part, the inspection will first find the left edge and then the bottom edge of the part. The intersection of these two edges will produce the bottom left coordinates that are required for the calibration of the system. The inspection will also check that the part used for calibration is the correct part by performing a match pattern. The match pattern will compare the taught part with the part located in the image. If the part does not match, then the smart camera will transmit a string indicating a calibration failure. A calibration failure can also occur if the inspection fails to locate the left or bottom edge. The smart camera will then transmit "*STX;Calibration;Failed;ETX*" message string to the Information Management System

indicating a failed inspection. If all the checks in the inspection are true, then the smart camera will transmit "*STX;Calibration;X:218;Y:39;ETX*" as an example. The x and y values indicate the coordinates by calculating the intersection of the left and bottom edge. The Information Management System will break up this message string into a string array in order to extract the x and y coordinates. The x and y coordinates will be saved in a SQL database and will be used in calculations when calculating the position that the x and y axis must move to in order to complete a positioning task accurately. The flow diagram of the calibration inspection can be seen in Figure 3.18.

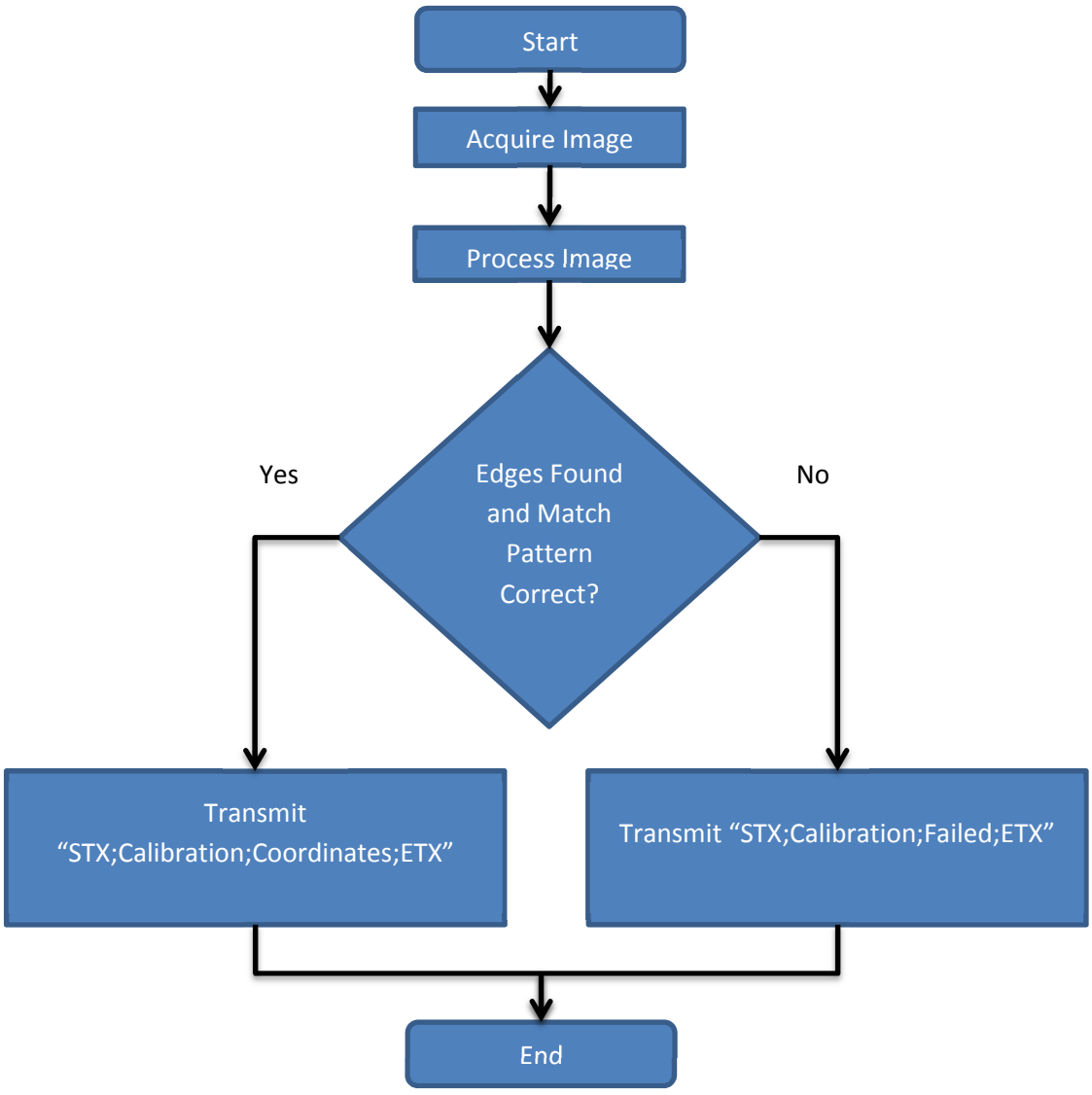


Figure 3.18: Flow diagram of Calibration inspection

3.4.3 Inspection Selection

The inspection selection is performed using the TCP/IP socket to send an inspection number to the smart camera from the Information Management System. The inspection number is assigned to an inspection name by the user using the VBAI software. The value assigned to the inspections is indicated in Table 8. The selection value indicated in the table is a string value, but the hexadecimal equivalent can also be sent to change the inspection. There is a program running in the background of the smart camera looking for the inspection selection value to be sent over the TCP/IP socket connection. The smart camera will change to another inspection if it receives a valid inspection selection value followed by a trigger from the PLC.

Table 8: Inspection Selection Values

Inspections	
Inspection Name	**Inspection Selection Value**
Find Task	1 (0x31)
Check Empty	2 (0x32)
Find Positions	3 (0x33)
Verify	4 (0x34)
Calibration	5 (0x35)

3.4.4 Smart Camera Agent

The Smart Camera Agent was designed using Microsoft Visual Studio 2008 and programmed using the .Net language of C#. Visual Studio was the perfect program to use for the development as it is a familiar development environment to me. C# was chosen as it is a programming language that I have used for many years and allowed me to develop the agent at a much faster rate than having to learn a new language, which would have added to the development time. The Smart Camera Agent is a static class that is used by the Information Management System to control and receive message strings from the smart

camera. The Smart Camera Agent provides the Information Management System with the following capabilities:

- Connect and Disconnect to the smart camera using TCP/IP sockets.
- Inform the Information Management System if the smart camera is connected or disconnected.
- Inspection selection by transmitting the selection value to the smart camera.
- Inform the Information Management System when new data has been transmitted by the smart camera by activating an event.
- Decode the message string transmitted by the smart camera.

The Smart Camera Agent was programmed in way that allows the sending and receiving of data to be done asynchronously. This means that methods that send and receive data will not block each other as they are run on different threads. The use of asynchronous methods also prevents the user interface from hanging when a read or write over the socket is occurring. An example of an asynchronous method for reading smart camera data is as follows:

```
private static void ConnectCallback(IAsyncResult result)
            {
                try
                {
                    Socket client = (Socket)result.AsyncState;
                    client.EndConnect(result);
                    ConnectedFeedback();
                        client.BeginReceive(buffer, 0, buffer.Length,
                        SocketFlags.None, new AsyncCallback(ReadCallBack),
                        client);
                }
                catch (Exception ex)
                {
                    Disconnect();
                    Error_Message = ex.ToString();
                }

            }
```

```
        private static void ReadCallBack(IAsyncResult result)
        {
            try
            {
                Socket client = (Socket)result.AsyncState;
                if (client != null)
                {
                    client.EndReceive(result);
                    int readCount = buffer.Length;

                    if (readCount > 0)
                    {
                        switch (buffer[0])
                        {
                            case 0x02:
                                {
                                    DecodeData(buffer);

                                    if (NewCameraData != null)
                                    {

                                  NewCameraData(null, new EventArgs());
                                    }
                                    break;
                                }
                        }
                    }

                 client.BeginReceive(buffer, 0, buffer.Length,
SocketFlags.None, new AsyncCallback(ReadCallBack), client);
                }
            }
            catch (Exception ex)
            {
                Disconnect();

            }


        }
```

The *ConnectCallback* method is called when the Information Management System and the smart camera establish a connection. This method then calls the *ReadCallBack* method which is executed asynchronously. The *ReadCallBack* method will check for data in the buffer that will then need to be checked for the STX (0x02) character. If the character is found, then the buffer will decode the data for the current inspection. The information Management System will be informed of the new data by calling the *NewCameraData* event. The *ReadCallBack* method will then be called again to listen for new data again. The *ReadCallBack* method essentially sits in an asynchronous loop listening for new data from the smart camera.

The transmission of data from the Information Management System to the smart camera is also performed using an asynchronous method although it will not sit in a continuous loop like the *ReadCallBack* method discussed above. The example below shows how the inspection can be changed from one to another using the send method.

```
public static bool ChangeInspection(int InspectionNumber)
          {
              try
              {
                  if (client != null)
                  {
                 Byte[] InspName =
                Encoding.ASCII.GetBytes(InspectionNumber.ToString());
                 client.BeginSend(InspName, 0, InspName.Length,
                SocketFlags.None, new AsyncCallback(Send), client);
                  }
              }
              catch (Exception ex)
              {
                  Disconnect();
                  Error_Message = ex.ToString();
                  return false;
              }
              return true;

          }
          private static void Send(IAsyncResult result)
          {
              try
              {
                  Socket client_conn = (Socket)result.AsyncState;
                  client_conn.EndSend(result);
              }
              catch (Exception ex)
              {
                  Disconnect();
                  Error_Message = ex.ToString();
              }
          }
```

The Information Management System will call the *ChangeInspection* method with a number of the inspection it requires. The integer value is converted to a byte value necessary for transmission. The socket will then call the asynchronous *Send* method with a variable containing the new inspection required. Once the send is complete then the *Send* method will end.

The Smart Camera Agent provides the Information Management System with the following methods and events as seen in Table 9.

Table 9: Smart Camera Agent methods and events

Smart Camera Agent	
Method	**Description**
Connect(string ip,int port)	Connect to camera with IP Address on port. Example: *Connect("192.168.27.150",4200)*
Disconnect	Disconnect from camera
ChangeInspection(int InspectionNumber)	Change camera inspection. Example: *ChangeInspection(2)*
Events	**Description**
CameraConnected	Event occurs when the camera connects to the Information Management System
CameraDisconnected	Event occurs when the camera disconnects from the Information Management System
NewCameraData	Event occurs when the camera transmits new data to the Information Management System

3.5 Mitsubishi PLC Ethernet Interface

3.5.1 Introduction

The interface with the Mitsubishi PLC control system has the ability to interface to external devices with the use of the Mitsubishi FX3U-ENET expansion module. This module allows the PLC to be connected with host systems, such as personal computers, and other PLC's using the TCP/IP or UDP/IP communication protocol via Ethernet [6]. The Ethernet interface provides the PLC with another programming interface and the ability to connect to HMI's and SCADA systems. The FX3U-ENET module is programmed using the FX Configurator-EN software. FX Configurator-EN supports the parameter-setting function to perform the Ethernet modules' initialization and the open processing with external devices [6]. A maximum of eight connections can be established and communicated with at the same time in the FX3U-ENET. The ability to have up to eight Ethernet connections allows the system to be expanded to incorporate additional devices in the event that the system needs to adapt

to any changes in the product. This factor is an important aspect within a RAS as the system can be expanded relatively easy to connect new devices if needed.

3.5.2 FX3U-ENET Interfacing Setup

All parameter settings and downloads for the FX3U-ENET module were performed using the FX Configurator-EN software. The software is user-friendly, making the parameter setup easy to understand. The Ethernet module IP Address has been set up to 192.168.27.101. Only three of the eight possible Ethernet connections have been used for this system. Figure 3.19 shows the three connections and the additional setting for each connection. The Ethernet module provides a dual functionality. The PLC can now be programmed through the Ethernet module and it allows for external devices to connect to the RAS. The module now provides the RAS system with a common communication medium.

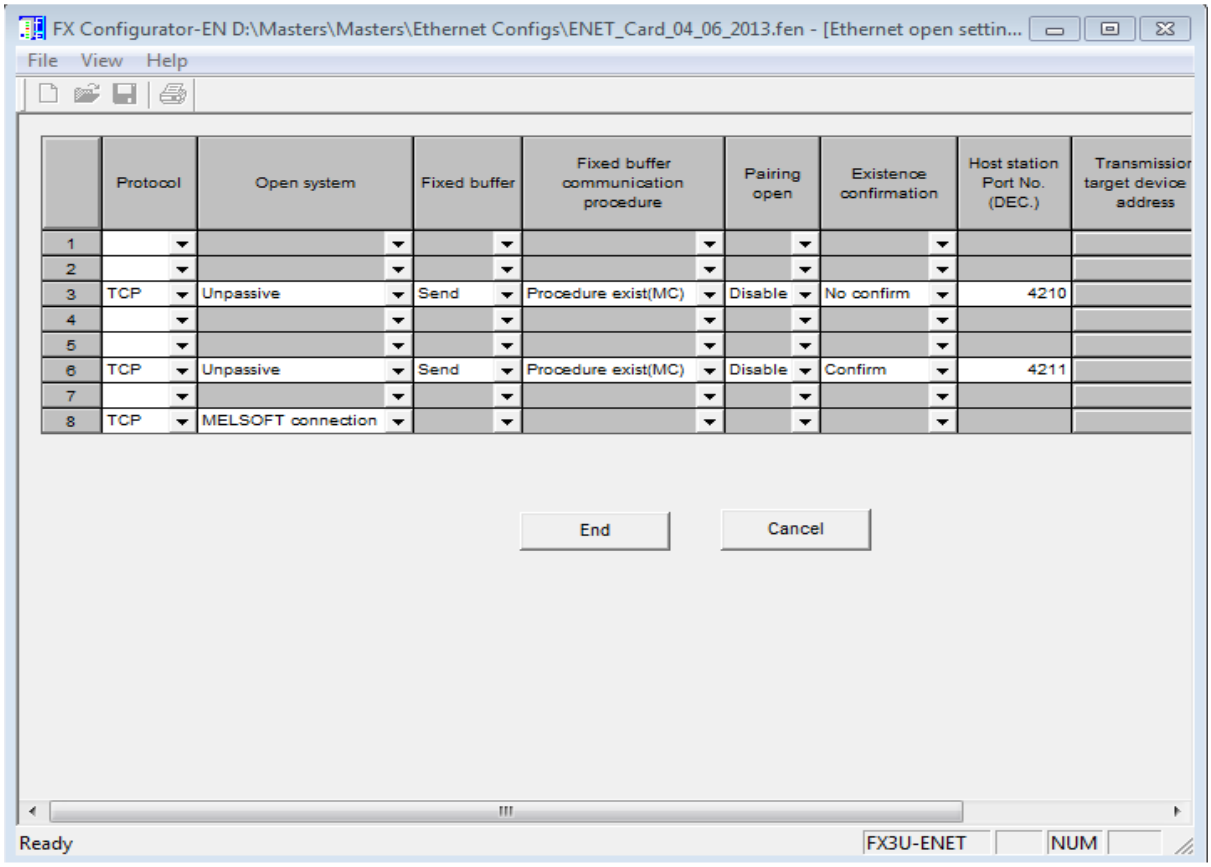


Figure 3.19: FX3U-ENET connection settings

Connection three of the Ethernet module will be used by the Information Management System to read/write PLC memory. This connection will allow the Information Management System to do the following:

- Trigger the smart camera
- Read motor controllers' status feedback
- Read alarm statuses
- Control the motor controllers for positioning tasks
- Start, Stop, Reset the system

Connection three acts as a TCP Server (Unpassive) and will listen on port 4210 for any connection requests from a TCP Client. The TCP Client in this case is the PLC Agent, which will be discussed in the next section. The communication procedure used is the "Procedure Exists (MC)". This procedure is used for the communication developed by Mitsubishi that allows external devices to communicate with the PLC via the Ethernet module. The MC protocol will be discussed in detail in the next section as a driver was written that forms part of the PLC Agent.

Connection six of the Ethernet module has been set up to allow additional devices to connect to the PLC on a separate connection to the Information Management System. This connection has been set up to be a TCP Server and listen for any TCP Client connection on port 4211. The main purpose of this connection is to give the RAS the ability to connect to mobile devices wirelessly to read/write PLC memory. This function allows operators to diagnose faults, correct faults, and control the RAS from a mobile device from within the factory without having to be in front of the RAS. Any mobile device can communicate with the Ethernet module, independent of its operating system, on condition that it acts as a TCP Client and uses the MC protocol. This opens up the RAS to communicate with Windows, Android and Apple devices.

Connection eight has been set up to allow for the programming of the PLC over the wired Ethernet connection or via the WIFI access point. This connection also allows for the online monitoring of the PLC code for any diagnostics or online changes that may need to be made.

The five unused connections can be used to add additional HMI's or a SCADA system in the future. The Ethernet module makes the RAS flexible if it needs to add additional devices to system with relatively quick software configuration, which will decrease the downtime of the system when changes to the product need to be made.

3.5.3 Mitsubishi PLC Agent

The Mitsubishi PLC Agent was designed using Microsoft Visual Studio 2008 and programmed using the .Net language of C#. This agent is very similar to the Smart Camera Agent in the way in which a connection is established, the connection is closed, the transmission of data and also the receiving of data. The message format used to communicate with the PLC Ethernet module is called MC protocol using A compatible 1E frames [6].

The Mitsubishi PLC Agent is a static class like the smart camera agent that is used by the Information Management System and a Windows Mobile device to control and receive message strings from the PLC Ethernet module. The Mitsubishi PLC Agent provides the Information Management System and a Windows Mobile device with the following capabilities:

- Connect and Disconnect to the Mitsubishi Ethernet module using TCP/IP sockets over a wired or wireless connection.
- Inform the Information Management System and the Windows Mobile device whether the PLC Ethernet module is connected or disconnected.
- Reading the status of the Festo stepper motor controllers and other status bits related to the operation of the RAS.
- Start, stop and reset the RAS.
- Trigger the smart camera inspection by sending a command to switch on the output to trigger the smart camera.

The MC protocol makes use of a command and response type of communication. For every command sent to the Ethernet module, a response message is expected. The response message could contain information related to the read command issued or it could just be a code indicating that a write command was executed correctly. Since the setup in the FX Configurator-EN was set to communicate using binary data, the protocol was written for the transmission of binary data and not ASCII data. Figure 3.20 shows an example of the command and response message format. It can be seen from Figure 3.20 that the command to read eight points of data starting from M100 is sent to the Ethernet module. The response code will return a complete code and the status of the eight bits from M100 through to M107.

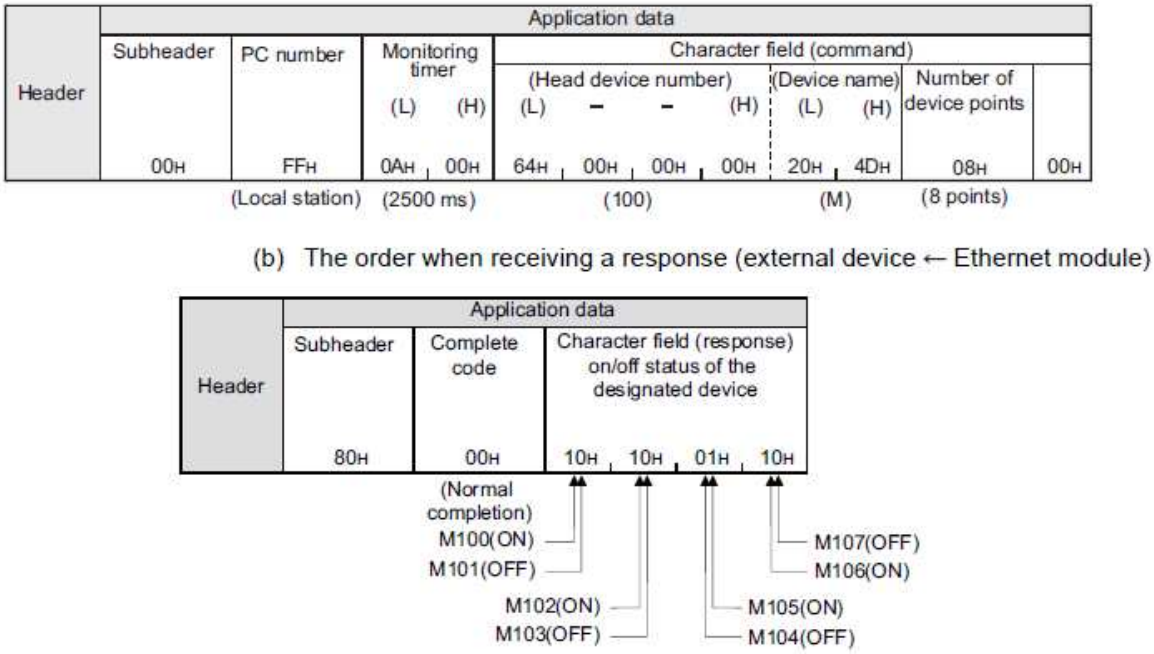


Figure 3.20: MC Protocol command and response format

The subheader determines the type of command that the Ethernet module must respond to. Figure 3.21 shows the various commands and the type of memory areas that can be used. The response subheader is the command subheader that is added to 80 Hex. This helps with identifying the type of command that was transmitted when trying to interpret the response data from the Ethernet module.

Item		Command/response type	Processing	Number of points processed per communication
Batch read	Bit units	00H	Reads bit devices (X, Y, M, S, T, C) in 1-point units.	256 points
	Word units	01H	Reads bit devices (X, Y, M, S, T, C) in 16-point units.	32 words (512 points)
			Reads word devices (D, R, T, C) in 1-point units.	64 points
Batch write	Bit units	02H	Writes to bit devices (X, Y, M, S, T, C) in 1-point units.	160 points
	Word units	03H	Writes to bit devices (X, Y, M. S, T, C) in 16-point units.	10 words (160 points)
			Writes to word devices (D, R, T, C) in 1-point units.	64 points
Test (random write)	Bit units	04H	Sets/resets bit devices (X, Y, M, S, T, C) in 1-point units by arbitrarily designating the devices and device number.	80 points
	Word units	05H	Sets/resets bit devices (X, Y, M, S, T, C) in 16-point units by arbitrarily designating the devices and device numbers.	10 words (160 points)
			Writes to word devices (D, R, T, C, etc.) in 1-point units by arbitrarily designating the devices and device numbers. Not applicable for 32 bit devices from C200 to C255.	10 points

Figure 3.21: Commands for reading and writing to the device memory

The Mitsubishi PLC Agent sends the data in the same format as seen it Figure 3.20. The command is transmitted asynchronously to the Ethernet module so as to not affect the Information Management System GUI (Graphical User Interface) and any other processes that are currently taking place. The example below shows how the Information Management System uses the Mitsubishi PLC Agent to read the status of memory bits M100 through to M111 to update the Information Management Systems GUI or even the Windows Mobile device applications GUI.

```
public static void Read()
{
    try
    {
        if (client != null)
        {
      Byte[] InspName = new byte[] { 0x00, 0xFF, 0x0A, 0x00,
      0x64, 0x00, 0x00, 0x00, 0x20, 0x4D, 0x0C, 0x00 };
       client.BeginSend(InspName, 0, InspName.Length,
       SocketFlags.None, new AsyncCallback(Send), client);
        }
    }
    catch (Exception ex)
    {
        Disconnect();
        Error_Message = ex.ToString();

    }

}
```

There are many other memory devices in the PLC that can also be read and written using these commands. The MC protocol provides access to all devices that could possibly be used in the PLC. Figure 3.22 shows the list of available devices and the device range.

Device		Device code	Device range	Device number
Data register		D (44H, 20H)	D0 to D7999	0000 to 1F3FH
			D8000 to D8511	1F40 to 213FH
Extension register		R (52H, 20H)	FX3G:R0 to R23999 FX3U/FX3UC:R0 to R32767	FX3G:0000 to 5DBFH FX3U/FX3UC:0000 to 7FFFH
Timer	Current value	TN (54H, 4EH)	FX3G:T0 to T319 FX3U/FX3UC:T0 to T511	FX3G:0000 to 013FH FX3U/FX3UC:0000 to 01FFH
	Contact	TS (54H, 53H)	FX3G:T0 to T319 FX3U/FX3UC:T0 to T511	FX3G:0000 to 013FH FX3U/FX3UC:0000 to 01FFH
Counter	Current value	CN (43H, 4EH)	C0 to C199	0000 to 00C7H
			C200 to C255	00C8 to 00FFH
	Contact	CS (43H, 53H)	C0 to C199	0000 to 00C7H
			C200 to C255	00C8 to 00FFH
Input		X (58H, 20H)	FX3G:X0 to X177 FX3U/FX3UC:X0 to X377	FX3G:0000 to 007FH FX3U/FX3UC:0000 to 00FFH
Output		Y (59H, 20H)	FX3G:Y0 to Y177 FX3U/FX3UC:Y0 to Y377	FX3G:0000 to 007FH FX3U/FX3UC:0000 to 00FFH
Internal relay		M (4DH, 20H)	M0 to M7679	0000 to 1DFFH
			M8000 to M8511	1F40 to 213FH
State		S (53H, 20H)	S0 to S4095	0000 to 0FFFH

Figure 3.22: Device codes and numbers used by the MC Protocol

All SCADA packages that list the Mitsubishi FX ENET device as a compatible device make use of this protocol to exchange data. Many other vendors will not show the communication protocol that is used to communicate with their device as it is considered proprietary information. There are many SCADA packages in the industry that have already written the driver based on the MC protocol, but the cost of buying a license and tags resulted in the development of the custom driver.

3.6 Information Management System

The Information Management System can be seen as the communicator in the system. It not only receives information from the devices of the RAS, but also controls the processes associated with the production of a product and the capabilities of the connected RAS. The Information Management System can be installed on any computer running Microsoft Windows XP or Microsoft Windows 7. All the data related to the products and the RAS are

stored in a Microsoft SQL Server 2008 R2 Express database. The following software and services were installed on one computer system:

- Information Management System written in C#
- Microsoft SQL Server 2008 R2 Express database
- Web Service written in C# and hosted by Microsoft's IIS (Internet Information Services)

It is possible to install all the software on different computers as the connection strings that connect to the SQL database and the web service can be modified to meet the required network configuration. For the purposes of this research project, all software mentioned above was installed on one Lenovo laptop with the following specifications:

- Processor: Intel® Core™ i7-2620M CPU @ 2.70 GHz.
- Installed RAM of 4.0 GB.
- Operating System: Windows 7, 32 bit.
- IP Address: 192.168.27.100
- Subnet Mask: 255.255.255.0

Figure 3.23 shows a breakdown of the individual software components of the computer and the communication between the different software and hardware components used in the RAS. Appendix B shows the flow diagram of the decisions made in the Information Management System when processing data from the smart camera.

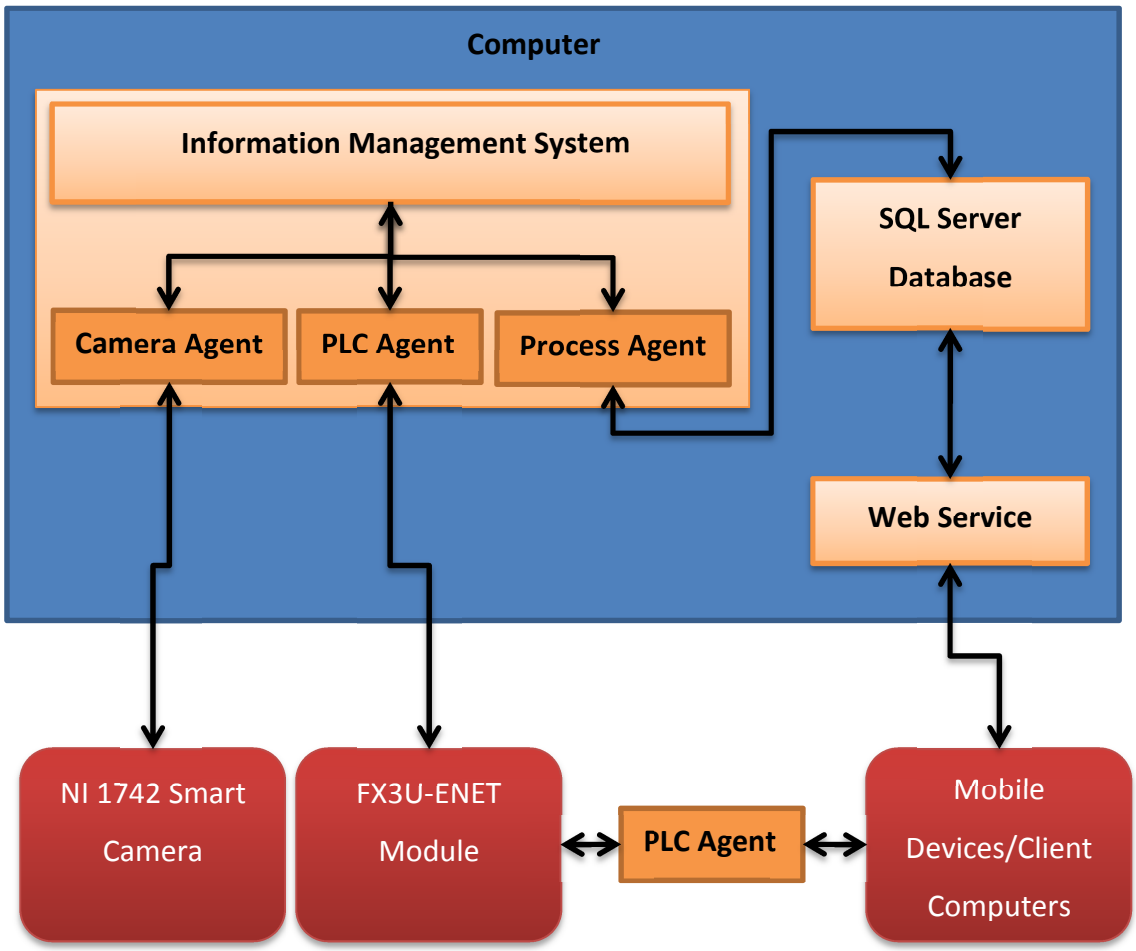


Figure 3.23: Individual software and hardware communication components

An in-depth discussion of the SQL Server database tables, stored procedures, Process Agent and Web Service will be presented in the sections to follow.

3.6.1 SQL Server Tables

The SQL Server database was implemented using Microsoft SQL Server 2008 R2 Express. This is a free database from Microsoft that does not require any licenses to run; however, there are limits when using the free version. SQL Server Express supports 1 physical processor, 1 GB memory and 10 GB storage [7]. These limitations do not in any way affect the performance of the RAS system, though. The SQL Server tables contain all information related to the RAS, products, tasks and logs. The next section will discuss the different table

purpose within the RAS and show examples of the data stored in the database. Appendix A reveals the table column name and data types for each table. All the tables discussed below are used by the Information Management System to carry out specific functions such as creating new products, adding orders and determining what task must be carried out by the RAS.

3.6.1.1 Axes Master Table

The Axes Master Table is used by the Information Management System to record the last time the x and y axes performed a homing operation. A homing operation is an important aspect that must be carried out regularly on the RAS as this determines the accuracy of the pick-and-place operations. The system has been set to home automatically when it has been powered up and also after each building task has been completed. Figure 3.24 shows an example of Axes Master Table data.

	AxisID	Axis	Description	HomedDT
▶	1	X	X Axis	2013-04-19 14:10:06.327
	2	Y	Y Axis	2013-04-19 14:08:01.297
*	NULL	NULL	NULL	NULL

Figure 3.24: Axes Master Table example data

The *HomedDT* field will be updated with a new date/time each time the x and y axes are homed. The x and y axes date/time stamp will not necessary be the same as the x and y axes could be at different positions when the homing operation is initiated. The update of the date/time field occurs when the axis reaches its home position and then notifies the Information Management System of a successful home operation.

3.6.1.2 Calibration Master Table

The Calibration Master table contains the x and y values of the axes as well as the x and y coordinates of the calibration part that was obtained using the Calibration inspection. A date/time value indicating the date is also stored. The table will record every calibration that was performed on the system. The Information Management System will, however, ignore all old calibration logs and only look for the most recent calibration date/time when performing the positioning calculations. Figure 3.25 shows an example of calibration logs stored in the Calibration Master table.

	CalibrationID	X_Axis_Value	Y_Axis_Value	X_Camera_Value	Y_Camera_Value	CalibrationDT
	1	341	277	118	112	2013-08-02 09:38:57.327
	2	442	203	232	35	2013-08-02 11:02:42.550
▶*	NULL	NULL	NULL	NULL	NULL	NULL

Figure 3.25: Calibration Master Table example data

3.6.1.3 Camera Master Table

The Camera Master Table stores the IP address and port number of the smart camera system. The information Management System will use this table to access the IP address and port details when attempting to connect to the smart camera with the Smart Camera Agent. All the camera details are shown in Figure 3.26.

	CameraID	CameraName	CameraDescription	CameraIP	CameraPort
	1	RAS Camera	Assembly Cell 1 Inspection Camera	192.168.27.150	4200
▶*	NULL	NULL	NULL	NULL	NULL

Figure 3.26: Camera Master Table example data

3.6.1.4 Comment Master Table

The Comment Master Table is a collection of possible events that could occur during the production of a product. The specific product can be tagged by adding a comment to the Job Master Table. This helps operators diagnose the reason for certain events or defect with the product. It is also used to check if a product has been built and verified before it leaves the production facility. This adds an element of quality control to the production line. Figure 3.27 shows a number of predefined comments loaded into the system.

	CommentID	CommentDescription
▶	1	Created
	2	Building
	3	Verifying
	4	Complete (Verified)
	5	Complete (Not Verified)
	6	Obstruction Detected
	7	Not All Positions Found
*	NULL	NULL

Figure 3.27: Comments Table predefine comments

3.6.1.5 Job Master Table

The Job Master Table contains all current jobs that must be processed or jobs that are currently being processed as well as jobs that have been processed in the past. The table also holds data of date/time events when the product was created, scanned by a RAS camera and when the build was determined to be completed. It also has a priority field for orders that are required to be completed first. Figure 3.28 shows an example of the Job Master table with three products that must be built by the RAS.

	JobID	ProductID	BuildID	TaskID	CommentID	CameraID	CreateDT	ScanDT	CompletionDT	Priority
	1	2	#5237400070027	3	1	NULL	2013-07-16 17:13:12.750	NULL	NULL	1
	2	3	#9856000002500	3	1	NULL	2013-07-16 17:15:01.977	NULL	NULL	1
	3	4	#3289400000020	3	1	NULL	2013-07-16 17:16:39.277	NULL	NULL	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 3.28: Job Master Table with job data

3.6.1.6 Job Log Table

The Job Log Table is a table that contains records of all transactions (insert, update, delete) that occurred on the Job Master Table. This table is used as a traceability log to track a specific product throughout production showing every event that occurred on it. This will also help it in tracking points in the production line where faults are occurring regularly. This table is a copy of the Job Master table with additional fields that track what type of transaction occurred on a product, which system made that change and which fields were modified. This table is updated by creating an SQL update, insert and delete trigger on the Job Master Table.

3.6.1.7 Part Master Table

The Part Master Table contains all parts that can be used in the production of a product and also the coordinates of the part location on the RAS build table. The Information Management System will use the location coordinates to move to the specific part when required. Figure 3.29 shows the Part Master Table with the three parts used in the products and the location coordinates of each. It also shows an empty part which will be ignored in positioning, but is used in the Product Master Table when creating pallets that require that some positions remain empty. The Information Management System has the ability to add, delete and modify part information if required.

	PartID	PartDescription	LocationX	LocationY
▶	1	Empty	0	0
	2	Solid Oval	250	40
	3	Hollow Oval	300	40
	4	Square	350	40
*	NULL	NULL	NULL	NULL

Figure 3.29: Parts Master Table with predefine parts and locations

3.6.1.8 PLC Master Table

The PLC Master Table, just like the Camera Master Table, holds the IP address and port number of the PLC system. The Information Management System will use this table to access the IP address and port details when attempting to connect to the PLC using the PLC Agent. All PLC details used in this research are shown in Figure 3.30.

	PLCID	PLCName	PLCDescription	PLCIP	PLCPort
▶	2	Mitsubishi PLC	Assembly Cell 1 PLC	192.168.27.101	4211
*	NULL	NULL	NULL	NULL	NULL

Figure 3.30: PLC Master Table data

3.6.1.9 Product Master Table

The Product Master Table holds information on all products that the RAS is able to build and also the type of parts that make up the product. The Information Management System will use the Product Master Table data to build the correct product. The data will also be used to compare the information that the smart camera detected and what product should have been built when the Verify Inspection is carried out. Figure 3.31 shows the three products that have been preloaded into the database. The Information Management System can add, delete and modify products.

	ProductID	ProductBarcode	ProductDescription	Position1	Position2	Position3	Position4	Position5	Position6	Position7	Position8	Position9	Position10	Position11	Position12	CreateDT
	2	#5237	Product 1	1	2	2	4	2	2	2	2	2	4	2	3	2013-07-16 17:11:54.987
	3	#9856	Product 2	4	1	2	3	3	2	4	1	2	3	3	3	2013-07-16 17:11:54.987
	4	#3289	Product 3	4	1	1	2	4	4	2	1	3	2	1	1	2013-07-16 17:11:54.987
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 3.31: Product Master Table product information

Position one through to position twelve of product ID two of Figure 3.26 indicates the part ID of the parts used in the Parts Master Table that should be placed in its respective position. Figure 3.32 shows the expected look of a base pallet when using Product ID two as an example.

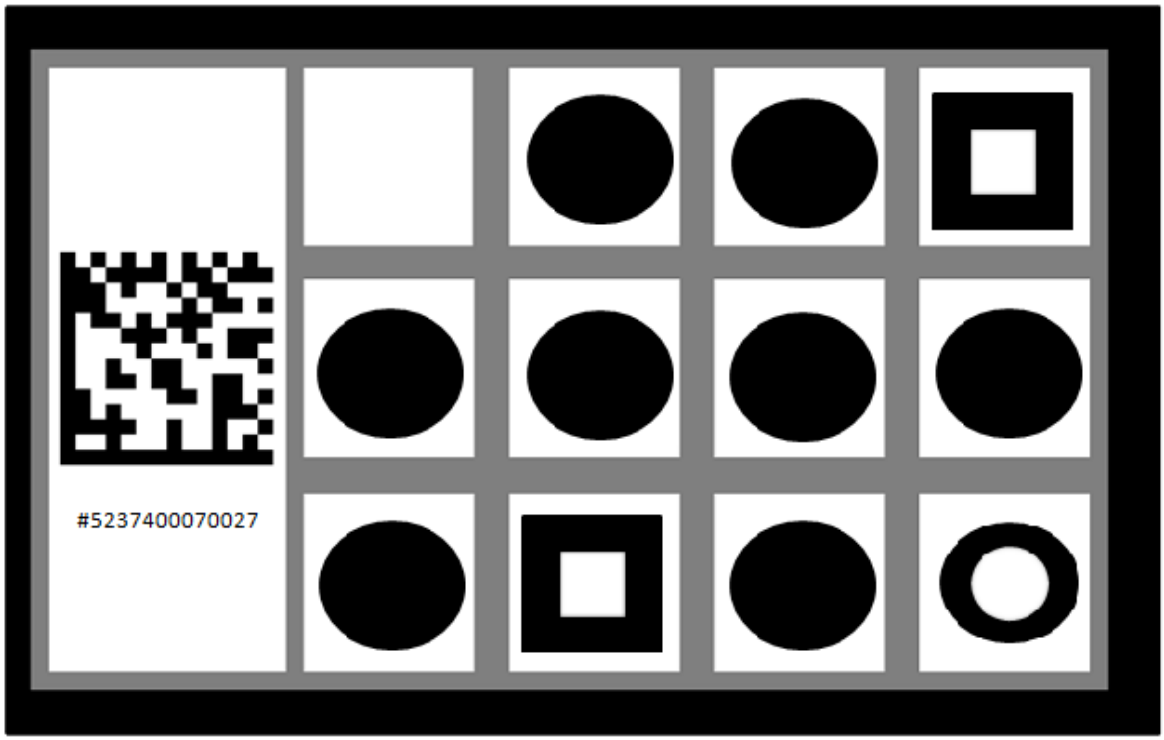


Figure 3.32: Product build configuration example

3.6.1.10 Task Master Table

The Task Master Table contains the different tasks that the RAS is able to perform depending on its physical capabilities. The Information Management System uses this information to decide on the steps that should take place when a base pallet is scanned at a RAS. Figure 3.33 shows the three different tasks that are available to the RAS.

	TaskID	TaskDescription
▶	1	Build Product
	2	Verify Product
	3	Build and Verify Product
*	NULL	NULL

Figure 3.33: Task Master Table data

The definition of the three tasks is as follows:

- Build Product – The product is required to be built without any verification from the RAS.
- Verify Product – The product only needs to be verified by the RAS. This could occur if the product was built manually and then placed on the system to verify the build.
- Build and Verify Product – The product is required to be built and verified before the build is considered complete.

3.6.2 SQL Stored Procedures

A number of stored procedures have been created that can select, update, insert and delete data that are related to the RAS. The stored procedures have been created using Microsoft SQL Server Management Studio 2008 R2. Management Studio provides a user-friendly user interface to view tables, and create functions, database triggers and - in this case - create stored procedures. The creation of stored procedures allows the code to be centralised in the database. This enables programmers to call a piece of code from the server instead of writing custom code to perform the same task for their application.

The stored procedures can be used in desktop applications, web applications and also web services, as discussed in the next section. This makes them platform independent and adds a layer of flexibility and modularity to the RAS at a software level. The introduction of a stored procedure to the RAS also allows the stored procedure to be modified while the system is running without affecting the system at all. The modified stored procedure will be updated after it has been compiled. Devices that are currently using the stored procedure under modification will not even notice a change. The stored procedures created during the development of the RAS can be used to do the following:

- Create, update and delete products.
- Read status information of the RAS.
- Read all events that occurred on a specific product throughout the production.
- Find jobs that must be processed at an assembly cell when requested from the Information Management System.

A simple example of a stored procedure used within the RAS shows how all the products in the "*Product_Master*" table are selected using the select command. This stored procedure can be modified easily to only select a few columns of the table if needed.

```
ALTER PROCEDURE [dbo].[usp_GetALLProducts]
      -- Add the parameters for the stored procedure here
AS
Begin
      -- SET NOCOUNT ON added to prevent extra result sets from
      -- interfering with SELECT statements.
SET NOCOUNT ON;
Select * from Product_Master

End
```

Using the stored procedures has proved to be very effective in creating reusable code that multiple applications can access without having to change the code to suit the various programming languages used today.

3.6.3 Web Service

The web service has been created using Microsoft Visual Studio 2008 and programmed in C#. The web service is hosted on the IIS (Internet Information Services) web server, which has been installed on a laptop connected to the RAS local network. A web service facilitates the sharing of information and services among companies and individuals on a programmatic level more elegantly than any other existing technology [8]. A web service can drastically reduce the development and integration time required to add additional functionality to a system. Another advantage of using a web service is that a web service can be consumed using many different programming languages such as Java, C++ and Visual Basic. This allows devices from different manufacturers running different operating systems the ability to communicate with the assembly system over a common architecture. The introduction of a web service to the RAS adds elements of reconfigurability, modularity and scalability.

The web service is relatively easy to implement and can be adjusted quickly to meet product changes and new functionality added to the RAS. The web service was created to illustrate a method of allowing different devices to communicate with the RAS through a common interface without the need to create device-specific applications. The web service adds the following function to the RAS:

- View all available products and product detail
- Add, delete and modify product configurations
- Track a specific product in real time
- View all events that occurred on a specific product
- View the status of the RAS and the status of the hardware components

All of the above functions can be accessed from mobile devices, tablets and PC's either through a website using a web browser or through a dedicated application created for the device. A Windows mobile application has been created to show the functionality of the web service using a mobile device. An example of a piece of code in the web service shows

how to select all the available products from the database of the RAS and return the results to the client application in a data set. The web service handles the connection to the SQL database and calls all the necessary stored procedures required for the method to return the correct results.

```
[WebMethod]
public DataSet GetAllProducts()

{
    string sqlconn = "Data Source=LYLE-PC\\SQLEXPRESS;Initial
    Catalog=Masters;Integrated Security=True";
    SqlConnection conn = new SqlConnection(sqlconn);
    SqlCommand cmd = new SqlCommand();
    cmd.CommandText = "usp_GetAllProducts";
    cmd.CommandType = CommandType.StoredProcedure;

    cmd.Connection = conn;
    conn.Open();

    SqlDataAdapter da = new SqlDataAdapter(cmd);
    DataSet productDS = new DataSet();

    da.Fill(productDS);

    conn.Close();
    return productDS;
}
```

This method can be called from any device that uses the web service as a web reference. In order for the client application to access the web service methods, a connection must first be established. The code below shows how a client subscribes to the web service, calls the "*GetAllProducts*" method, stores the result in a dataset and displays the data set in a combo box control.

```
//Subscribe to the web service
 RAS_WebService.RAS_webservice ras_ws = new
 RAS_WM_App.RAS_WebService.RAS_webservice();

 //Create a data set
 DataSet dsProducts = new DataSet();

 //Call the "GetAllProducts" and store the result in the
dsProducts dataset
 dsProducts = ras_ws.GetAllProducts();

 //Display the products in a combo box for later selection
 cmb_AllProducts.DisplayMember = "ProductDescription";
 cmb_AllProducts.DataSource=dsProducts.Tables[0].DefaultView;
```

3.6.4 Windows Mobile Application

The Windows Mobile application was produced using Microsoft Visual Studio 2008 and programmed using C#. The Windows Mobile device used to test the application was a HTC TYTN 2 mobile phone running Windows Mobile 6. The application can also run on many other devices running Windows Mobile as well as Windows CE. The introduction of mobile devices into automation systems is increasing as the applications developed allow users to view real-time data on the system from anywhere in the world. The application developed is able to communicate with the Information Management System and the assembly cells PLC. The method in which the communication takes place is described below. Figure 3.34 shows an example of a screen that shows the product number and the product design.

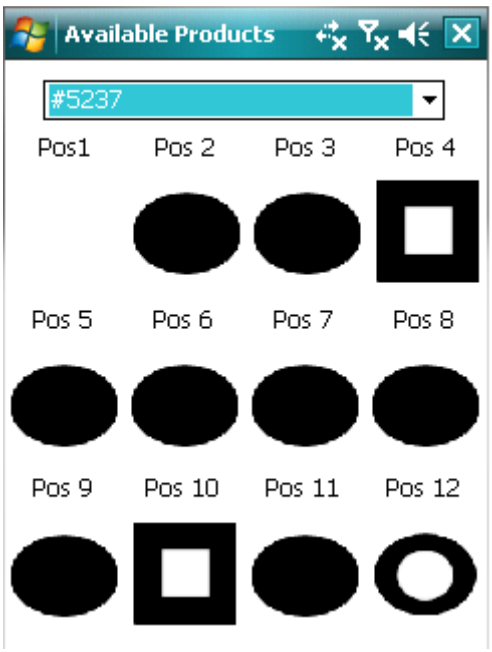


Figure 3.34: Available products screen from the mobile application

The physical communication between the application and the Information Management System is done over the wireless network by connecting the mobile device to the access point. The application will make a connection to the web service when a transaction is requested by the user of the mobile device. The web service will perform the requested

transaction and return the requested data in the correct format needed to display on the mobile device. The web service provides the application with the following functionality:

- View all available products and product detail
- Add, delete and modify product configurations
- Track a specific product in real time
- View all events that occurred on a specific product
- View the status of the RAS and the status of the hardware components.

The physical communication between the application and the assembly cell PLC uses the same medium of communication as the one describe above. The application however makes a direct connection to the PLC without using the web service. The application creates a connection to the PLC Ethernet module on connection six using port 4211 as discussed in section 3.5.2. The connection is created using the PLC Agent to communicate as the driver between the PLC and the mobile application. The application is able to provide the user with the following functions through the PLC Agent:

- Connect and Disconnect to the Mitsubishi Ethernet module using TCP/IP sockets over a wireless connection.
- Inform Windows Mobile device if the PLC Ethernet module is connected or disconnected over the wireless network.
- Reading the status of the Festo stepper motor controllers and other status bits related to the operation of the RAS.
- Start, stop and reset the RAS.

The addition of the mobile application is to show the flexibility and modularity that the PLC agent and web service brings to the system without having to make any changes. The web service required no changes to be used with the mobile application. The only change required for the PLC Agent was that it had to be compiled for the Windows Mobile program to use the .NET Compact Framework. Although the compact framework does not always

provide the same functions that are available to Windows desktop applications, the PLC Agent was designed to work with both desktop and mobile devices.

3.7 References

[1] [Online] Available from http://www.festo.com/cms/en-za_za/14894.htm, [Accessed 18 June 2012]

[2] [Online] Available from http://www.ni.com/pdf/products/us/cat_ni_1742.pdf [Accessed 29 May 2012]

[3] [Online] Available from http://www.festo.com/net/SupportPortal/Files/52020/fhpp_5_10parameter [Accessed 29 May 2012]

[4] Mitsubishi Electric, 2011/2012, *'The Automation Book'.*

[5] [Online] Available from http://www.datamatrixcode.net/ [Accessed 30 June 2013]

[6] Mitsubishi Electric November 2009, *'FX3U-ENET User Manual'.*

[7] [Online] Available from http://www.microsoft.com/enza/download/details.aspx?id=30438 [Accessed 30 June 2013]

[8] Fiach Reid, *Network Programming in .NET with C# and Visual Basic .NET*, Elsevier Digital Press, United States of America.

Chapter 4: Results

4.1 Introduction

This chapter discusses the tests performed on the vision system and the building of a product. There were many factors that could influence the vision system's accuracy. Examples are ambient lighting and placement of product parts.

4.2 Test Methods and Analysis

4.2.1 Benchmark Inspections

To establish how fast each inspection functioned, benchmark tests were performed on all the inspections consisting of ten samples. The benchmark test results are shown in Table 10. The average inspection time was obtained using VBAI built-in benchmarking application.

Table 10: Benchmark Inspections

Inspections		
Inspection Name	**Samples**	**Average Inspection Time**
Calibration	10	180.058 ms
Find Task	10	551.398 ms
Check Empty	10	818.991 ms
Find Positions	10	3297.119 ms
Verify	10	1969.400 ms

Each inspection was set up in the following manner:

- Calibration Inspection – The calibration indicator was placed in the vision systems field of view and then the calibration inspection was set up to perform ten inspections. In order for the inspection to be considered valid, a valid x and y coordinate must be returned over the TCP/IP connection for these ten inspections.

- Find Task Inspection – A valid barcode that meets the criteria for a barcode discussed in section 3.2.2.2.1 was placed in the barcode section of the product base pallet. The inspection was set up to perform ten inspections. All ten inspections returned the correct barcode information successfully.
- Check Empty Inspection – This test was performed by making sure that all parts were removed from the base pallet. The inspection was set up to perform ten inspections. All ten inspections identified the twelve base pallet positions as empty.
- Find Positions Inspection - This test was performed by making sure that all parts were removed from the base pallet. The inspection was then set up to perform ten inspections. All inspections returned valid x and y coordinates for the twelve positions of the base pallet.
- Verify Inspection – This test was performed by placing a combination of the three parts onto the twelve position of the base pallet. Each one of the twelve positions contained a part. Ten inspections were performed and each inspection was checked against the actual part in the position against what the vision system determined to be in that position. All ten inspections return 100% verification results.

4.2.2 Inspection Accuracy Tests

All these tests were used to check the accuracy and repeatability of the individual inspections. A number of samples were obtained using different methods depending on the inspection type.

4.2.2.1 Calibration Tests

This test was performed to determine the average deviation and average of the calibrated x and y coordinates that are transmitted from the vision system over the TCP/IP connection to the Information Management System during calibration of the x and y axes with the vision system.

The following steps were carried out to obtain control coordinates for the test:

- The calibration part was placed in the vision system's field of view.
- The calibration inspection was run in order to get the x and y coordinates from the vision system. The x and y coordinates was used as the control for all future inspections for this test.
- The X-Axis and Y-Axis were homed.
- The X-Axis and Y-Axis were moved so that the Z-Axis was centred with the bottom left hand point of the calibration part. The x and y coordinates of the X-Axis and Y-Axis was recorded and used as the control for all future inspections for this test.
- The X-Axis and Y-Axis was homed again.

This test consisted of running the inspection to retrieve the x and y coordinates of the calibration part, calculate the position that the X-Axis and Y-Axis should move to and then send the position to the motor controller. The calculated position and the actual position of the axes were then compared to find any error. The results of the test can be seen in Table 11.

Equation (4.1) and equation (4.2) is used to calculate the "Calculate Axes Position" value for X and Y respectively.

Calculated Axes Position(x) = Camera Position(x) + Control (X-Axis Actual) – Control (Camera Position(x) (4.1)

Calculated Axes Position(y) = Camera Position(y) + Control (Y-Axis Actual) – Control (Camera Position(y) (4.2)

Since the X and Y axes have encoders and are used in a closed loop, the actual position of the axes will move to the calculated value and maintain that position until another position is given. This resulted in a 0% error on all tests as the axes moved to the correct position. There was, however, a slight variation in the camera position coordinates. This is a result of the camera only able to send out whole numbers and not decimal numbers so it rounds up or rounds down the camera coordinates.

Table 11: Calibration Test Results

Calibration Tests							
Test Number	**Camera Position**		**Calculated Axes Position**		**Actual Axes Position**		**% Error (Calculated Axes Position versus Actual Axes Position)**
	X (mm)	**Y (mm)**	**X (mm)**	**Y (mm)**	**X (mm)**	**Y (mm)**	
Control	**219**	**38**			**442**	**202**	
1	219	38	442	202	442	202	0%
2	219	38	442	202	442	202	0%
3	219	38	442	202	442	202	0%
4	219	38	442	202	442	202	0%
5	219	38	442	202	442	202	0%
6	220	38	443	202	443	202	0%
7	219	38	442	202	442	202	0%
8	219	38	442	202	442	202	0%
9	219	38	442	202	442	202	0%
10	219	39	442	203	442	203	0%
Average Deviation	**0.18 mm**	**0.18 mm**					
Average	**219.1 mm**	**38.1 mm**					

4.2.2.2 Find Task Test

This test was performed to determine the accuracy of the vision system when scanning Data Matrix barcodes at different times of the day. The barcode data were then transmitted from the vision system over the TCP/IP connection to the Information Management System.

The tests were conducted over the course of the day; these were performed to test the repeatability of the inspection with changing ambient lighting as a factor. A Data Matrix barcode was placed on the base pallet in the area where a barcode should be placed. Each test conducted resulted in obtaining 10 samples from the vision system. The test results are shown in Table 12.

Table 12: Find Task Results

Barcode Inspection Tests			
Test Number	Samples	Incorrect	% Correct
1	10	0	100%
2	10	0	100%
3	10	0	100%
4	10	0	100%
5	10	0	100%
6	10	0	100%
7	10	0	100%
8	10	0	100%
9	10	0	100%
10	10	0	100%

All one hundred samples resulted in the vision system reading the barcode correctly and transmitting the barcode data to the Information Management System in the correct format.

4.2.2.3 Check Empty Test

This test was performed to determine the accuracy of the vision system when checking the base pallet for the twelve empty positions and then transmitting the result of the inspection back to the Information Management System over the TCP/IP connection.

The tests were conducted over the course of the day to test the inspection under different ambient lighting conditions. All twelve positions of the base pallet were cleared of any parts. Each sample of the test conducted resulted in running the inspection and then reading the result in the Information Management System. Each test involved ten samples. The results of this test can be seen in Table 13.

Table 13: Check Empty Inspection

Check Empty Inspection Tests			
Test Number	Samples	Incorrect	% Correct
1	10	0	100%
2	10	0	100%
3	10	0	100%
4	10	0	100%
5	10	0	100%
6	10	0	100%
7	10	0	100%
8	10	0	100%
9	10	0	100%
10	10	0	100%

All one hundred samples collected resulted in the vision system correctly identifying that the pallet was empty. All samples also transmitted the correct data to the Information Management System.

4.2.3.4 Find Position Tests

This test was performed to determine the accuracy of the vision system when finding the twelve positions of the base pallet and then transmitting the result of the inspection back to the Information Management System over the TCP/IP connection.

The tests were conducted over the course of the day and were used to test the repeatability of the inspection with the changing ambient lighting as a factor. An empty base pallet was placed in the view of the smart camera and the camera was triggered to take an inspection using the VBAI software. Ten samples were taken and the x and y coordinates was obtained for each position and then tabulated. Appendix C contains the results obtained from this test.

An average deviation was calculated to determine the variation in the x and y position for each inspection. The deviation on some of the samples could have been caused by the

lighting as the tests were conducted in a room that was affected by ambient lighting. All position coordinates were successfully transmitted to the Information Management System through the TCP/IP connection without any problem.

4.2.3.5 Verify Inspection Tests

This test was performed to determine the accuracy of the vision system when verifying the part located in the base pallet's positions. The position and the part in that position were transmitted to the Information Management System over the TCP/IP connection.

These tests, just like the others, were conducted throughout the day in order to test the repeatability with changing ambient lighting. A base pallet was placed in the view of the smart camera and all twelve positions of the base pallet either contained a part or remained empty. Each test run consisted of ten samples and a different part configuration. Table 14 shows the results obtained from the test.

Table 14: Verify Inspection Results

Verify Inspection Tests			
Test Number	**Samples**	**Incorrect**	**% Correct**
1	**10**	**0**	**100%**
2	**10**	**0**	**100%**
3	**10**	**0**	**100%**
4	**10**	**0**	**100%**
5	**10**	**5**	**50%**
6	**10**	**0**	**100%**
7	**10**	**0**	**100%**
8	**10**	**1**	**90%**
9	**10**	**0**	**100%**
10	**10**	**0**	**100%**

Not all of the tests proved to be accurate. The two tests that failed to produce a 100 % correct part identification consisted of parts that had flaws in them. This resulted in the vision system not recognising the part as a result of the threshold being set too high for the

pattern matching function of the inspection. The adjustment of the threshold value helped in solving the problem with the other test runs until a reasonable threshold was set for a specific part. Another factor that contributed to the errors was once again an ambient lighting issue.

4.3 Introduction of new products

The introduction of new products into the system was done using the Information Management System. The Information Management Systems makes use of a GUI that enables users to add new products to the system. New products can also be added through the Windows mobile application created for the RAS. Figure 4.1 shows the GUI used to add new products to the RAS.

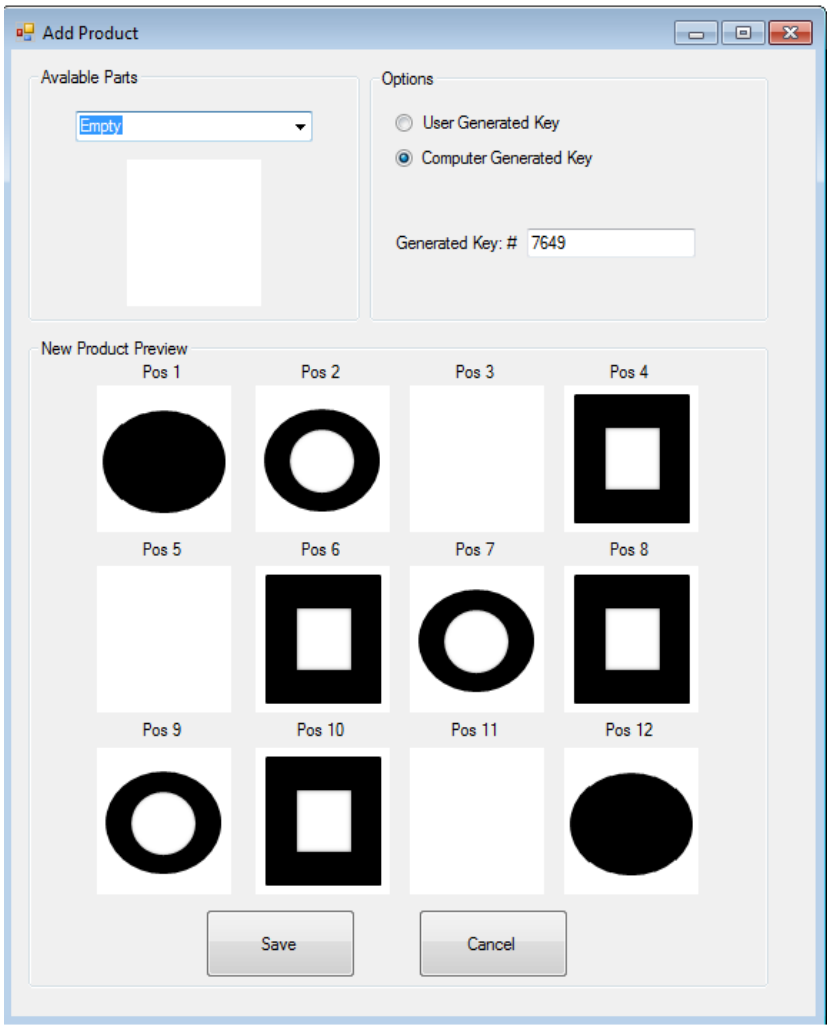


Figure 4.1: Add new product GUI

New products added to the system can either be assigned a user-defined unique key for the product or one can be automatically generated by the Information Management System. The unique key will always start with a # followed by four numeric characters. The "#" character was appended to the unique key string in the stored procedure. The unique key was generated using a random number function in the Information Management System and passed to the stored procedure with other parameters necessary in the introduction of new products. The unique key had to be tested against all other keys in the database to determine if a new unique key needed to be generated if one already existed.

A new job was created for the new product to test if the new product could be built without having to modify any hardware and software. An empty base pallet was placed on the build table with a barcode representing the product and build codes generated by the

Information Management System. The RAS was able to identify the product barcode successfully and began the steps in building and then verifying the final built product. All steps in the building and verification of the product were successful and no modification in terms of hardware or software was required. In the event that a new product range required a new part, other than the ones that have been used for testing, it was necessary to modify the vision systems inspections, the Smart Camera Agent to decode an extra part and the Information Management System to be able to display the part on verification status from the vision system.

Future research could be conducted into solving this problem on the side of the Information Management System. It will be relatively difficult to solve the problem in the vision system as it needs to learn the new part and the threshold setting also needs to be set for the part.

4.4 Conclusion

All the tests conducted above were used to check a number of important factors when producing an automated system; namely, accuracy, performance, reliability, repeatability and the ease of introducing changes into the system.

The Cartesian system proved to be very accurate as it has been used in a closed-loop system with the encoder providing feedback of the current position at all times. Although the Cartesian system was accurate in moving to a position sent from the Information Management System, the actual position in respect to the part to be picked up could differ by a few millimetres. The reason for this is that it relies on the operator lining up both the x and y axis with the calibration part in order for the Information Management System to learn the link between the vision system's coordinates and the Cartesian system's coordinates. In my view, a calibration tool should be applied to the x and y axes when performing calibration in order to remove the operator error, which will result in producing an accurate positioning system.

The performance of the vision system seems acceptable in that the average inspection time for the Verify Inspection was 1969.400 ms. Although two seconds might sound like a long

time in a production environment, one must take into account the amount of image processing taking place on the vision systems to verify which one of the three available parts are located in that position. It is possible to improve on this time by trying different techniques in identifying parts through the VBAI software. The vision system proved to be reliable and provided good accuracy as seen from the results in the Find Task, Check Empty, Find Positions and Verify tests.

The ease of introducing changes into the system did involve making changes to the Smart Camera Agent as well as the smart camera's Verify Inspection to recognise the new part introduced into the system. Creating new product variations that consisted of parts known to the smart camera inspection and Smart Camera Agent proved to be easy and required no software changes except for registering the new product with the system.

Attempting to find a solution to the manufacturing industry's reconfigurable problem is a tough task. Although the Smart Camera Agent and the PLC Agent used a common communication protocol (TCP/IP), the structure of the data transferred differs between devices, which results in creating changes to the agents to interpret the data specific to the device it is connected too.

Chapter 5: Conclusion and Recommendations

This dissertation documents the research conducted into the different software and hardware components used to create a Reconfigurable Assembly System (RAS). The research tended to focus more on the software development side and the different communication protocols available that would aid in achieving a reconfigurable system.

It took many months of research before a decision was made on the hardware and software that would be used for the RAS. Various software programs were written to test the communication protocols of all the devices to find a suitable method to obtain data in an efficient way. Another important factor was creating software that is generic enough to use with different vendors' equipment with little or no changes. A good example of the generic software developed during this research is the Smart Camera Agent and the PLC Agent. The software for connecting, disconnecting, writing data and reading data is exactly the same for the smart camera and the PLC. The only exception is the way in which the data must be interpreted for each device.

5.1 Contribution of this project

- The development of a Reconfigurable Assembly System with an Integrated Information Management System was successfully implemented using different hardware and software components that are able to communicate with each other.
- The PLC Agent was created in order to allow a Mitsubishi PLC to communicate with the Information Management System. A Smart Camera Agent was created to allow a vision system that has the ability to run as a TCP Server to communicate with the Information Management System. These two agents have not only been used for this project, but are currently being used in the manufacturing sector on automation projects that I have developed. The agents have also being expanded to work with other equipment besides PLC's and vision systems. Only minimal changes to the

agents were required, to have them communicating with the other devices in a short period of time.

- An Information Management System was produced to manage the agents, view system information and store system information in a database.
- A Windows Mobile application has also been developed that was based on some of the functions found in the Information Management System.
- The web service was also created and allows other applications to perform certain functions, such as viewing system data. Items like the Information Management System, Windows Mobile application and the web service have also been used in the manufacturing industry on many of my automation projects. The Windows Mobile application was an important addition to the dissertation as it not only shows the use of mobile devices in the manufacturing sector, but is following the current trend whereby the use of mobile devices in all aspects of life are growing at an extremely fast rate.

Over the last year, there has been an increased demand from customers wanting to integrate control systems with their Enterprise Resource Planning (ERP) system. The software created has the ability to communicate with the ERP systems and pass information between the control system and the ERP system. The software developed for the integration of the different hardware components is flexible, modular and can easily be adjusted to work with other devices that share a similar communication protocol.

5.2 Recommendations for future research

From the experience gained during this dissertation and the experience gained by working in various manufacturing industries, the following recommendations for further work are made:

- The RAS should be developed further to include a Z-Axis and a rotating gripper. A rotating gripper will add to the system's reconfigurability and accuracy.

- Further research into the integration of commonly used ERP (e.g. SAP, Manhattan) systems with the RAS for scheduling jobs. The majority of companies have an existing ERP system and are reluctant in changing the way the business operates.
- Different tools can be used to pick different objects, which would make the RAS flexible when it comes to adding more complex parts to the product line-up.

The knowledge gained during this study will aid in the development of Reconfigurable Assembly Systems within the manufacturing industry in South Africa. Creating an assembly system that is able to adapt to frequent market changes, with minimal changes to both software and hardware, will help in preventing some South African companies from moving their assembly operations to other countries due to the high cost and/or low productivity of labour. Reconfigurable Assembly Systems will also allow South African companies to compete with the global market in producing quality products.

Appendix A – Database Table Design

Table 15: Axes Master Table Design

Axes Master		
Column Name	**Data Type**	**Allow Nulls**
AxisID (PK)	smallint	Unchecked
Axis	varchar(1)	Unchecked
Description	varchar(250)	Checked
HomedDT	datetime	Checked

Table 16: Calibration Master Table Design

Calibration Master		
Column Name	**Data Type**	**Allow Nulls**
CalibrationID	bigint	Unchecked
X_Axis_Value	varchar(3)	Unchecked
Y_Axis_Value	varchar(3)	Unchecked
X_Camera_Value	varchar(3)	Unchecked
Y_Camera_Value	varchar(3)	Unchecked
CalibrationDT	datetime	Unchecked

Table 17: Camera Master Table Design

Camera Master		
Column Name	**Data Type**	**Allow Nulls**
CameraID	smallint	Unchecked
CameraName	varchar(50)	Unchecked
CameraDescription	varchar(250)	Checked
CameraIP	varchar(20)	Unchecked
CameraPort	varchar(10)	Unchecked

Table 18: Comment Master Table Design

Comment Master		
Column Name	**Data Type**	**Allow Nulls**
CommentID	smallint	Unchecked
CommentDescription	varchar(250)	Unchecked

Table 19: Job Log Table Design

Job Log		
Column Name	**Data Type**	**Allow Nulls**
JobID	bigint	Checked
ProductID	smallint	Checked
BuildID	varchar(14)	Checked
TaskID	smallint	Checked
CommentID	smallint	Checked
CameraID	smallint	Checked
CreateDT	datetime	Checked
ScanDT	datetime	Checked
CompletionDT	datetime	Checked
Priority	int	Checked
SPID	smallint	Checked
NT_USERNAME	varchar(128)	Checked
NT_DOMAIN	varchar(128)	Checked
HOSTNAME	varchar(128)	Checked
PROGRAMNAME	varchar(128)	Checked
LOGINNAME	varchar(128)	Checked
Operation	varchar(6)	Checked
ArchiveInd	tinyint	Unchecked
Date_Time_Logged	datetime	Unchecked

Table 20: Job Master Table Design

Job Master		
Column Name	**Data Type**	**Allow Nulls**
JobID	bigint	Unchecked
ProductID	smallint	Unchecked
BuildID	varchar(14)	Unchecked
TaskID	smallint	Unchecked
CommentID	smallint	Unchecked
CameraID	smallint	Checked
CreateDT	datetime	Unchecked
ScanDT	datetime	Checked
CompletionDT	datetime	Checked
Priority	int	Checked

Table 21: Part Master Table Design

Part Master		
Column Name	**Data Type**	**Allow Nulls**
PartID	smallint	Unchecked
PartDescription	varchar(250)	Unchecked
LocationX	int	Checked
LocationY	int	Checked

Table 22: PLC Master Table Design

PLC Master		
Column Name	**Data Type**	**Allow Nulls**
PLCID	smallint	Unchecked
PLCName	varchar(50)	Unchecked
PLCDescription	varchar(250)	Checked
PLCIP	varchar(20)	Unchecked
PLCPort	varchar(10)	Unchecked

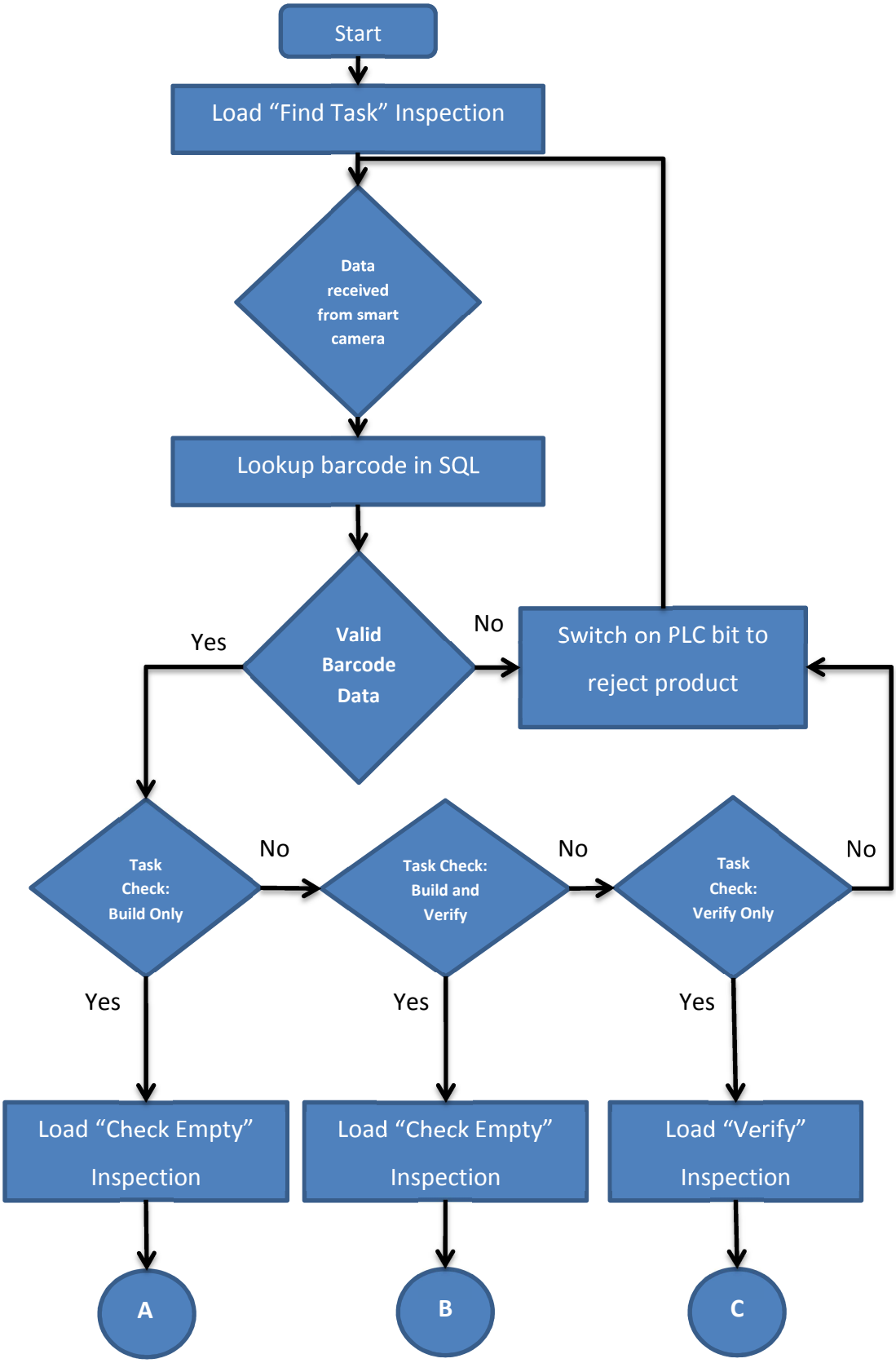
Table 23: Product Master Table Design

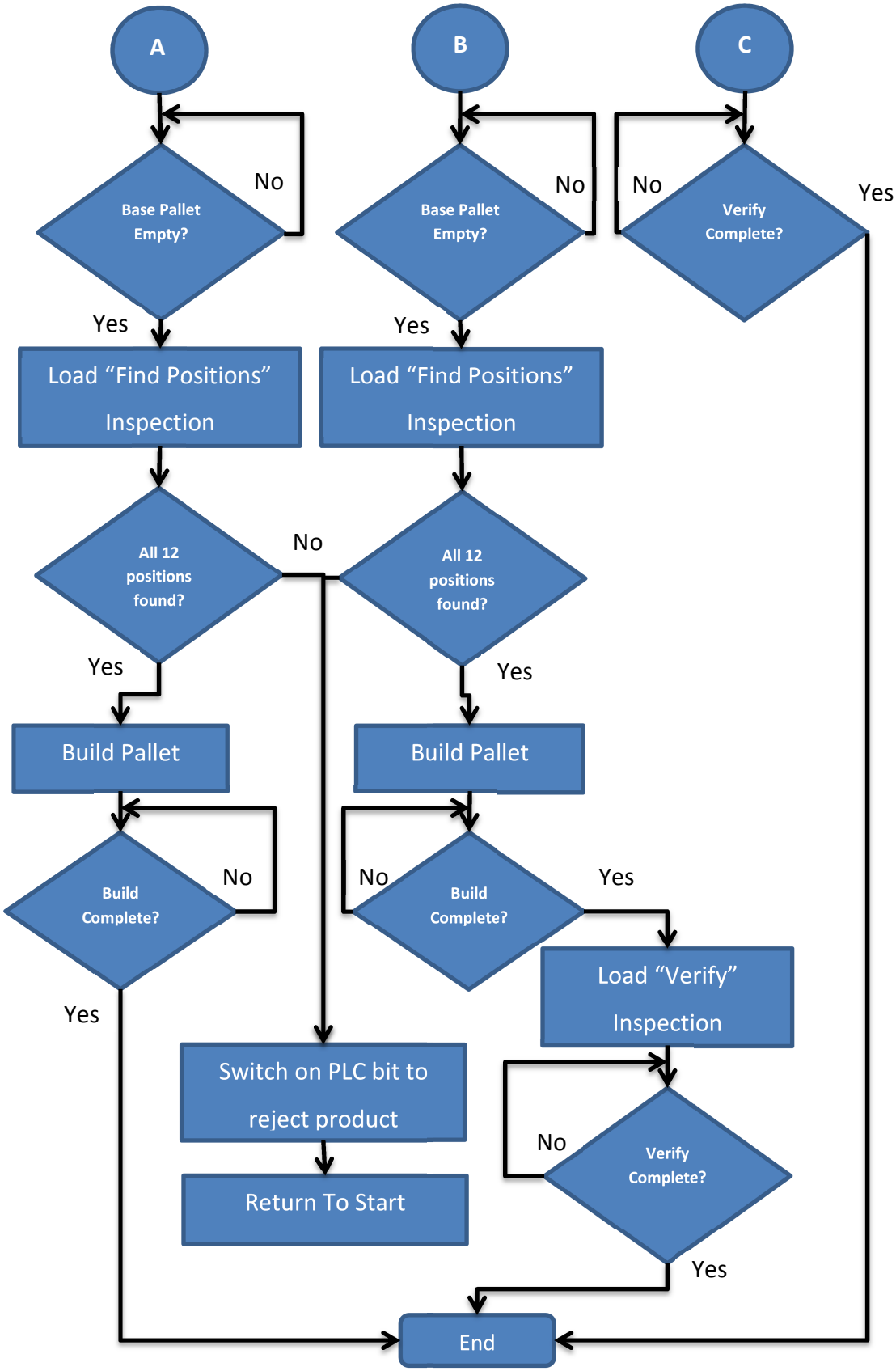
Product Master		
Column Name	**Data Type**	**Allow Nulls**
ProductID	bigint	Unchecked
ProductBarcode	varchar(5)	Unchecked
ProductDescription	varchar(250)	Checked
Position1	smallint	Unchecked
Position2	smallint	Unchecked
Position3	smallint	Unchecked
Position4	smallint	Unchecked
Position5	smallint	Unchecked
Position6	smallint	Unchecked
Position7	smallint	Unchecked
Position8	smallint	Unchecked
Position9	smallint	Unchecked
Position10	smallint	Unchecked
Position11	smallint	Unchecked
Position12	smallint	Unchecked
CreateDT	datetime	Unchecked

Table 24: Task Master Table Design

Task Master		
Column Name	**Data Type**	**Allow Nulls**
TaskID	smallint	Unchecked
TaskDescription	varchar(250)	Checked

Appendix B – Information Management System job handling




A
B
C
Base Pallet Empty?
No
Base Pallet Empty?
No
No
Verify Complete?
Yes
Yes
Yes
Load "Find Positions" Inspection
Load "Find Positions" Inspection
All 12 positions found?
No
All 12 positions found?
Yes
Yes
Build Pallet
Build Pallet
Build Complete?
No
No
Build Complete?
Yes
Load "Verify" Inspection
Yes
Switch on PLC bit to reject product
Return To Start
No
Verify Complete?
Yes
End

Appendix C – Find Positions Test Results

Find Positions Inspection Tests

Test Number	Samples	Position 1		Position 2		Position 3		Position 4		Position 5		Position 6		Position 7		Position 8		Position 9		Position 10		Position 11		Position 12	
		X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y
1	1	131	90	167	91	203	90	238	90	131	128	167	127	202	127	238	127	131	165	167	165	203	164	238	164
2	1	131	91	167	90	203	90	238	91	131	128	167	127	203	127	238	127	131	165	166	165	202	165	238	164
3	1	131	91	167	91	203	90	238	91	131	128	167	127	203	127	238	127	131	165	167	164	202	165	238	164
4	1	131	91	167	91	202	91	238	90	131	128	167	127	202	127	238	127	131	165	167	164	202	165	238	164
5	1	131	91	167	91	202	91	238	90	131	128	167	127	203	127	238	127	131	165	167	164	202	165	238	164
6	1	131	91	167	91	202	91	238	90	131	128	167	127	202	127	238	127	131	165	167	165	202	164	238	164
7	1	131	91	167	91	202	91	238	90	131	128	167	127	203	127	238	127	131	165	167	164	202	165	238	164
8	1	131	91	167	91	202	91	238	90	131	128	167	127	203	127	238	127	131	165	167	164	202	164	238	164
9	1	131	91	167	91	203	90	238	90	131	128	167	127	202	127	238	127	131	165	167	165	202	165	238	164
10	1	131	91	167	91	202	91	238	90	131	128	167	127	203	127	238	127	131	165	167	165	202	164	238	164
	Average Deviation	0	0.18	0	0.18	0.48	0.48	0	0.32	0	0	0	0	0.48	0	0	0	0	0	0.18	0.5	0.18	0.48	0	0
	Average	131	90.9	167	90.9	202.4	90.6	238	90.2	131	128	167	127	202.6	127	238	127	131	165	167	164.5	202.1	164.6	238	164