Development of a Hybrid Control and Monitoring System within a Reconfigurable Assembly System

By:

JOHAN ADAM NIEMANN

Thesis submitted in fulfilment of the requirements for the degree:

DOCTOR OF ENGINEERING: ELECTRICAL ENGINEERING

in the

Department of Electrical, Electronic and Computer Engineering

of the

Faculty of Engineering and Information Technology

at the

Central University of Technology, Free State

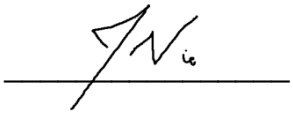
Supervisor:
Prof. H.J. VERMAAK (Ph.D.)

Co-supervisor:
Dr. N.J. LUWES (D.TECH)

Bloemfontein
2016

Declaration

I, JOHAN NIEMANN, do hereby declare that this research project which has been submitted to the Central University of Technology for the degree DOCTOR OF ENGINEERING : ELECTRICAL ENGINEERING, is my own independent work; and complies with the Code of Academic Integrity, as well as other relevant policies, procedures, rules and regulations of the Central University of Technology; and has not been submitted before by any person in fulfilment (or partial fulfilment) of the requirements for the attainment of any qualification.



Student Signature: Date: 25/11/2016

Acknowledgements

I would like to sincerely and deeply thank the following individuals and institutes, whom without the completion of this thesis would not have been possible:

- Firstly, to my creator The God Almighty, for giving me the determination, perseverance and capability to complete this project and thesis.
- To my spouse Nicolette, for your patience, understanding and loving support.
- To my parents, for raising me to be the best I can be, for all the opportunities you provided me and your unconditional love and support.
- To my study leaders, Prof. HJ Vermaak and Dr. NJ Luwes, for your guidance, endless knowledge and wisdom, and your friendship during the course of my studies.
- To CUT and RGEMS, for granting me the opportunity to undertake this project, for the monetary assistance, and the knowledge and experience gained during the course of the project.
- To my fellow research students within the RGEMS research group, I appreciate it that I could be part of the team (furniture). Thanks for your friendship, collaborations and knowledge sharing.

Abstract

Expanding global markets are constantly changing and unstable. South African manufacturing companies need to develop similar levels of sophistication and expertise in the automation industry as its international rivals, to compete for these markets and meet rising consumer expectations. To remain competitive, these manufacturing companies must manage their plants extremely efficiently to ensure the quality of assembled products; allow for rapid product introduction and product changes; achieve shortened throughput cycles; ensure more reliable delivery dates; and effectively coordinate product demand while contending with decreased product lifespans. To accomplish this, manufacturing companies in SA are progressively engaging in the current trend in automation known as reconfigurable manufacturing. Due to the extreme flexibility of these reconfigurable systems, the monitor and control systems for these require the same levels of flexibility. The purpose of the study is to develop a hybrid control and monitoring system, to supervise and control reconfigurable assembly systems (RAS), and adapt to the flexibility of these systems. To achieve this, a literature study was done in the research area to reveal the prerequisites for such systems; the physical assembly devices were designed and built; the separate software modules developed and ultimately integrated into the intended system. The tests to validate the system were developed in such a way that each subsection of the system is validated by using a different system software function. This inevitably confirms the functionality of the fundamental components and the system in entirety. The results indicated that devices are easily added to the system; devices are successfully detected and identified; how the system plans production, and how the system automatically configures itself. Further results showed the capability of the system to generate and virtually wire system runtime code; store and retrieve production data; as well as warn and alarm on unwanted conditions. By obtaining these results, companies can configure their systems with ease, in a shorter amount of time, and without any human error. Moreover, their systems will be more flexible, allow easy addition of new products and assembly devices, and with minimal downtime. This will enable SA manufacturing companies to be more competitive, ensure increased productivity, achieve extreme system flexibility, and decrease lead times – thus ensuring them an advantage over their international competitors.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

APS	Advanced Planning and Scheduling
CSV	Comma Separated Value
CUT	Central University of Technology
DIM	Detection and Identification Module
DMS	Dedicated Manufacturing System
DSC	Data Logging and Supervisory Control
ERP	Enterprise Resource Planning
FMS	Flexible Manufacturing System
GUI	Graphical User Interface
HCoMS	Hybrid Control and Monitoring System
HMI	Human Machine Interface
ID	Identification
IDE	Integrated Development Environment
IED	Intelligent Electronic Device
IO	Input/output
IP	Internet Protocol
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
LAN	Local Area Network
LCD	Liquid Cristal Display
LED	Light Emitting Diode
MES	Manufacturing Execution System
MMI	Man Machine Interface
MRP	Material Requirements Planning
MRP II	Manufacturing Resource Planning
NI	National Instruments
OPC	OLE (Object Linking and Embedding) for Process Control
PAC	Process Automation Controller
PC	Personal Computer
PCN	Process Control Network
PDF	Portable Document Format
PLC	Programmable Logic Controller
PSP	Publish-Subscribe Protocol
RAS	Reconfigurable Assembly System
RFID	Radio Frequency Identification
RGEMS	Research Group in Evolvable Manufacturing Systems

RMS	Reconfigurable Manufacturing System
RTU	Remote Terminal Unit
SA	South Africa
SCADA	Supervisory Control and Data Acquisition
SV	Shared Variable
SVE	Shared Variable Engine
UI	User Interface
USB	Universal Serial Bus
VI	Virtual Instrument
WAN	Wide Area Network

Chapter 1 Introduction to Study Environment

1.1 Introduction

Manufacturing companies today need to be highly sophisticated to compete for global markets and keep up with current worldwide manufacturing trends. Manufacturing companies are compelled to adopt and implement new methods in manufacturing to adapt to frequently and unpredictably changing markets to remain competitive. These market changes include frequent introduction of new products, changes in the product demand and fluctuations in batch orders [1]. As global markets demand a wider variety of products with consistent high quality, at a competitively low price, and delivered in the shortest possible time manufacturers are required to ensure top quality of their products and increase production throughput to meet these market demands [2]. In this regard, the systems that manufacturers utilize must be capable of high-volume production throughput, rapid change-over between products and quick ramp-up to full production potential. Such systems are required to be extremely flexible in both system structure and system capabilities.

This gives rise to the concept of reconfigurable assembly systems (RAS). However, these systems must be expertly monitored and controlled; and have reconfiguration abilities. In this respect, an intelligent supervisory control and monitoring system will ensure the successful control of such systems, improve production efficiency, increase system capabilities and, most importantly, increase product quality. Various sources in the literature show that extensive research has been done in the RAS field of study [3-6]; however, research regarding the monitor and control systems of RAS is limited and/or lacking. With this in mind, this research study focuses on the development of a Hybrid Control and Monitoring System (HCoMS), which must be as flexible and easily adaptable as the RASs it must monitor and control.

1.2 Problem Statement

Reconfigurable assembly systems need to be extremely flexible to function competitively in unstable global markets; due to this flexibility, the monitoring and control systems for these are not easily implemented.

1.3 Research Goals and Objectives

1.3.1 Hypothesis

South African manufacturing companies can be more competitive; increase productivity and product variety; decrease lead times; and provide product production information by utilizing hybrid control and monitoring systems developed for reconfigurable assembly systems.

1.3.2 Specific Objectives

The monitoring and control of RAS are not easily implemented due to the complexities of such systems. These complexities accumulate due to the extraordinarily flexibility of RAS; the nature of the hardware and software used in these systems; the infinite possibilities of distinct product designs; and the requirement to ensure the quality of these assembled products. In addition to these complexities, unpredictable markets require these systems to rapidly change and adapt to remain competitive. To accomplish this, the objectives of this study are to:

- Design and develop a hybrid control and monitoring system (HCoMS) for reconfigurable assembly systems.
- Design and develop the hardware to test the system concept.
  - The system must be able to adapt automatically, depending on the requirements of a specific product to be assembled.
  - The system must allow a user to manually configure the system as desired.
  - The system must be able to provide product production information.
  - The system must be able to ensure the quality of assembled products.
- Design and develop the user interface to be intuitive and conform to high internationally practised graphical user interface (GUI) standards.
- Design and develop the visual interface to be compatible with touch panels, tablets and smart phones.

1.4 Research Methodology

Reconfigurable assembly systems are highly flexible and complex systems, as are the monitoring and control systems that supervise and control it. Therefore, a Hybrid Control and Monitoring System (HCoMS) will be developed to supervise and control a reconfigurable assembly system that comprises various assembly processes. The

software platform of choice on which the HCoMS will be developed will be LabVIEW (Laboratory Virtual Instrument Engineering Workbench) from National Instruments (NI). LabVIEW is a powerful environment wherein visual front panel user interfaces (UI) can be created and complex background processing to control instruments can be implemented. LabVIEW allows for easy integration with various hardware systems and devices. In addition, a physical system will be constructed to validate the HCoMS.

The physical structure and design of the HCoMS will be implemented as shown in Figure 1.1. A high specifications panel computer with LabVIEW installed on it will have the main HCoMS application residing on it and execute the runtime processes of the system. It will be connected to a database server, OPC server, multiple machine vision stations and various modular assembly devices through an Ethernet or Wi-Fi network. An HCoMS must perform supervisory control over assembly devices and utilize the machine vision stations to ensure the precision of the assembly processes and the quality of the assembled products. Furthermore, the HCoMS will communicate with the database server to store the acquired production data and request required information from memory for control and reporting purposes.

To improve production control capabilities, a production planner will be implemented in the HCoMS. The production planner must access product-assembly recipes, compare recipes with the parts inventory, determine the required processes, allocate resources to these processes and schedule when these processes must occur. In addition, capabilities to extend the visual interface, by providing the ability to add human machine interfaces (HMI) and touch screen panels to the system, as well as providing the option to monitor the system using mobile devices such as smart phones and tablets, will be implemented. The UI running on the system will be developed with users in mind, must be easily comprehended, and meet high UI design standards. Furthermore, physical modular assembly devices will be constructed as sections of a possible system. These assembly devices will be added, removed and reorganized in different arrangements to change the system structure and test the capability of the HCoMS to adapt to these changes.

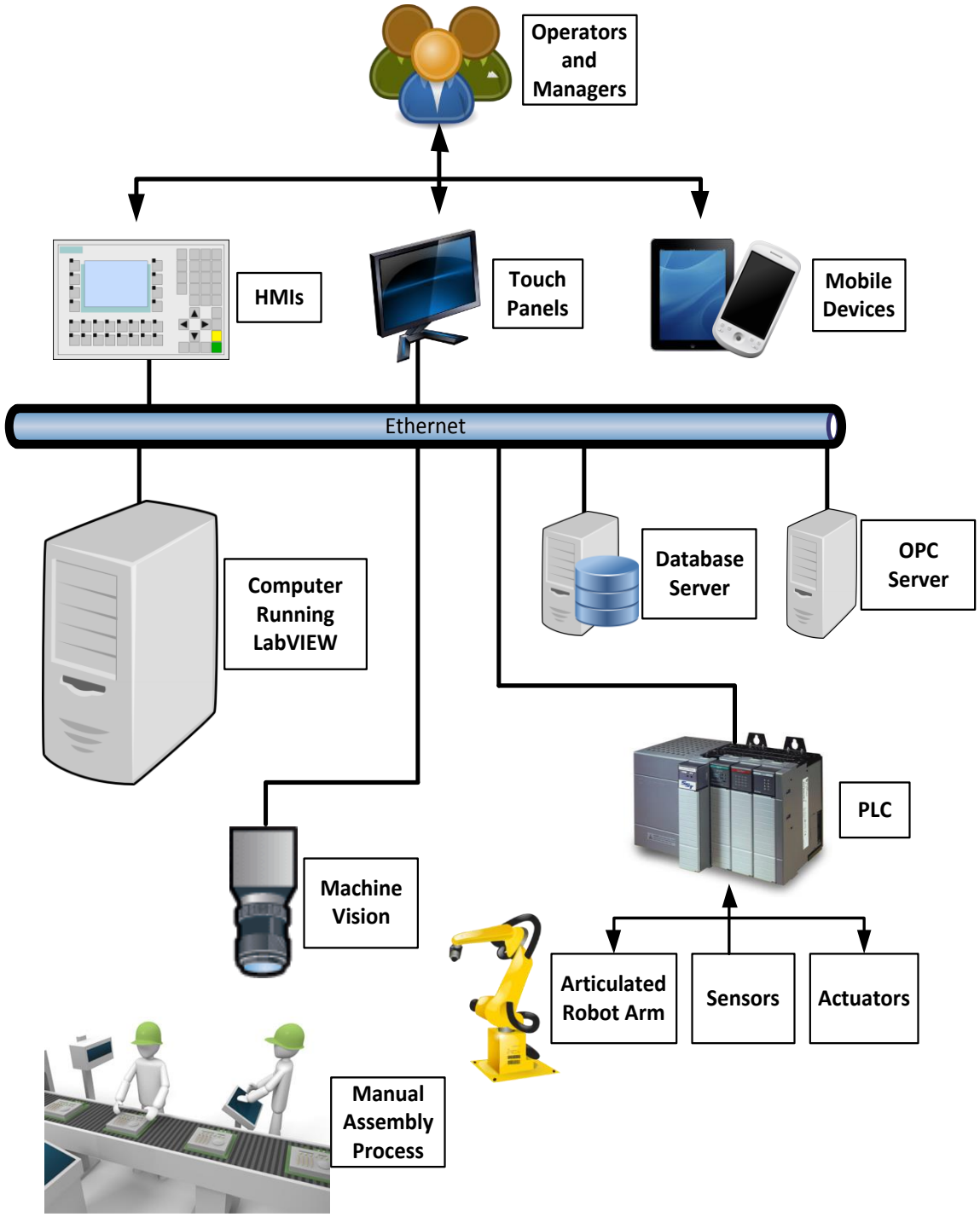


Figure 1.1 Proposed system layout overview

1.5 Layout of Thesis

Chapter 1: This chapter gives an introduction of the research problem; contains the problem statement and hypothesis; and highlights the objectives and research methodology of the project.

Chapter 2: This is a literature study dedicated to acquiring knowledge in the relevant field of study.

Chapter 3: This chapter reveals the methodologies undertaken to develop the system under study. These methodologies include identifying the system hardware and software components; a description of how to implement these components; and how these components integrate. All considered, this chapter provides the methods to develop the system in entirety.

Chapter 4: This chapter identifies the tests that need to be performed to verify the system. It also identifies the setup of each test, the procedures to perform each test and what results to expect.

Chapter 5: This chapter is a discussion on the results obtained from the tests done. It provides analysis on each test by comparing expected to obtained results; identifies corrections and improvements in each case; and includes additional results obtained during the project that were not part of the testing procedures.

Chapter 6: A concluding chapter to discuss the achievements of the project; contributions to industry; and future work to be completed.

Chapter 2 Theoretic Perspectives of Reconfigurable Systems

2.1 Introduction

This chapter consist of a literature review of the field of study. The literature study illuminate areas like reconfigurable assembly systems; supervisory control and data acquisition; HMI screen design; and manufacturing planning and scheduling systems. In addition, the study also includes an explanation on utilizing the LabVIEW development environment, as well as the LabVIEW DSC Module. In brief, this chapter provides current knowledge in the field of study and includes current practices utilized in industry.

2.2 Reconfigurable Assembly Systems

2.2.1 Introduction

Traditional systems utilized by the manufacturing industries in the product assembly environment include manual assembly systems, dedicated manufacturing systems (DMS) and flexible manufacturing systems (FMS). Manual assembly is mostly utilized in cases where products are of a complex nature; have multiple dimensional variations within a product family; require added manipulation for these variations; and are easily justifiable in countries where wages are low and unemployment high. DMS are special-purpose lines that are mainly justified when production volumes are high; the products do not have any variations; and the markets are stable. On the contrary, FMS are utilized when production volumes are relatively lower compared to DMS; more than one variation in product exist; or multiple variations of products are assembled simultaneously on the same line [7, 8].

Choosing the most suitable assembly system for a given manufacturing process can be a difficult and daunting decision, since there are several requirements to consider. Considerations regarding the product include product lifetime, dimensions, geometry, number of parts and number of variants. In addition, the management of manufacturing facilities expects high through-put at reasonable, low cost. Furthermore, the chosen type of assembly system also influences the productivity, quantity, flexibility and the diversity of variants. However, there are a variety of concepts to meet these requirements and assembly systems can generally be divided into one of three areas of utilization. These utilization areas are

manual assembly, hybrid assembly and automated assembly. By referring to Figure 2.1, manual assembly proves to have higher diversity of products and flexibility in assembly processes; while production rates and quantities are low. In contrast, this is the inverse for fully automated systems. It then becomes evident that by compromising on high quantities and production rates, the implementation of hybrid assembly provides higher flexibility and diversity, and still accomplish moderate productivity and quantities [9-12].

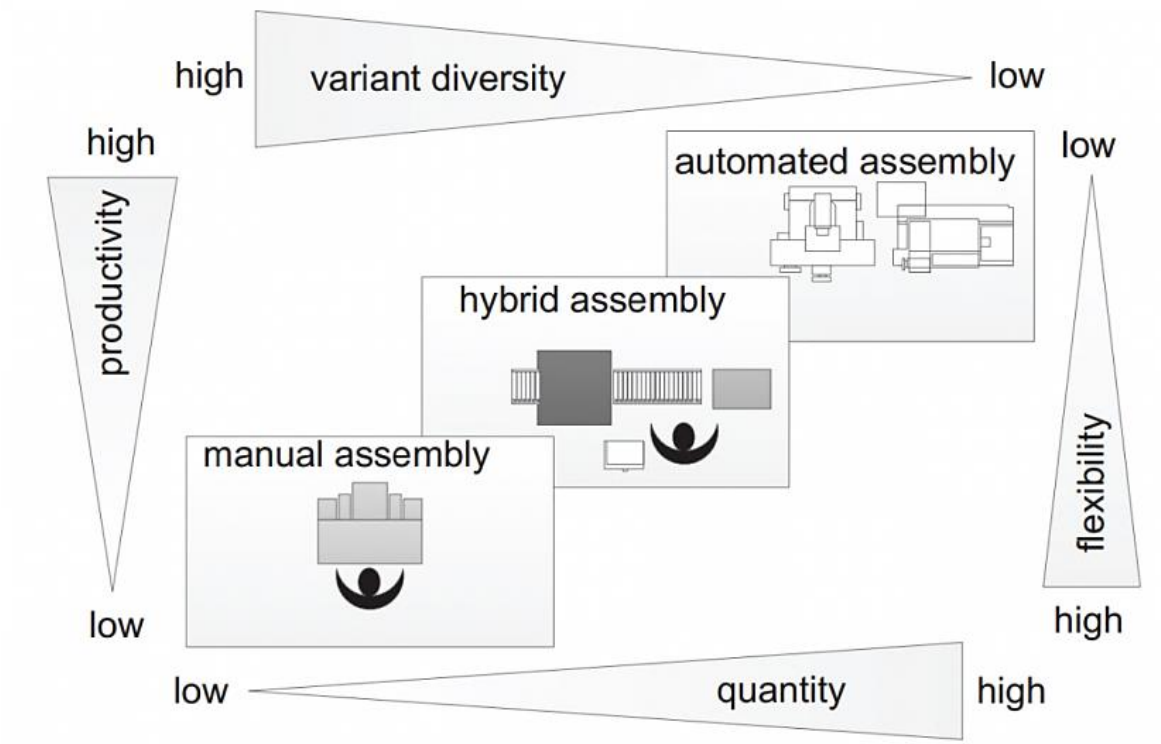


Figure 2.1 Assembly areas of utilization [11, 13]

To remain competitive, manufacturing companies use these assembly strategies to remain relevant in a market that is prone to frequent, unpredictable change. These market changes include the frequent introduction of new products, changes in the product and fluctuations in product demand and batch orders [14]. As the world market demands a wider variety of products at constant high quality, at competitively low prices and in the shortest time possible, manufacturers must ensure the quality of their products, and up productivity to compete for these markets [15]. To assist manufacturers in being more competitive, the systems utilized must be capable of high-volume production, rapid part change-over and rapid ramp-up to full production. Such a system must show a certain degree of flexibility with respect to

the system structure and the system capabilities. This gives rise to the concept of reconfigurable assembly systems.

2.2.2 Reconfigurability and Flexibility

Reconfigurability is defined as the operative ability to repeatedly change and rearrange the components of a system in a cost-effective way, through the addition or removal of functional elements with minimal effort and delay [16, 17]. In contrast, flexibility is the tactical ability of a production and logistics area to switch to a new, though similar, family of products by changing manufacturing processes, material flows and logistical functions within reasonably little time and effort. Furthermore, the key difference between reconfigurability and flexibility can be explained by the following:

- Firstly, the diversity of work pieces handled: reconfigurable systems may switch between different families of products, while flexible systems switch between similar products.
- Secondly, the extent of change that the manufacturing system has to undergo: reconfigurable systems may add or remove machine components, while flexible systems change the process or material flow.

Moreover, there are two types of reconfiguration that can occur in manufacturing systems, namely basic (or classic) and dynamic reconfiguration. Basic reconfiguration can be achieved by stopping the system, applying the necessary hardware or software changes, and then restarting the system. This is also known as “cold starting” the system. Dynamic reconfiguration is reconfiguration achieved during real-time operation of the system, without requiring to stop the system [18]. In addition to reconfigurability, the flexibility of manufacturing systems can be identified in different areas and are as follows [17, 19]:

- Machine: Various operations can be performed without set-up change.
- Material handling: Paths available for transfer of materials between machines.
- Operation: Various operation plans are available for part processing.
- Process: Different sets of part types can be produced without setup changes.
- Product: Ease of introducing products into an existing product range.

- Routing: The ratio of the number of feasible routes to the number of part types.
- Volume: The ability to vary production volume within production capacity.
- Expansion: Ease of increasing capacity through physical changes to the system.
- Control Program: The ability of a system to run virtually uninterruptedly due to the availability of intelligent machines and system control software.
- Production: The number of all part types that can be produced without adding major capital equipment.

2.2.3 Definition and Characteristics of Reconfigurable Assembly Systems

The concept of reconfigurable manufacturing systems are defined by Koren et al. [14, 20] as: *"a manufacturing system are designed at the outset for rapid change in structure, as well as in hardware and software components, in order to quickly adjust production capacity and functionality within a part family in response to sudden changes in market or regulatory requirements."* For a RMS to conform to the aforementioned, it must adhere to the following characteristics [19, 20]:

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

2.2.4 Reconfigurable Machines

Reconfigurable machines are designed to allow customized flexibility in a cost-effective manner. The machines are specially designed to handle variations within a specific product family. The degree of flexibility of a reconfigurable machine depends largely on its modularity. A modular design simplifies the changeover

procedure and provide the possibility of “plug and produce”. To design reconfigurable machines to conform to the characteristics above, the following design principles must be applied [21]:

- Design around a specific part family.
- Design for customized flexibility.
- Design for easy and rapid convertibility.
- Design for scalability; allow for addition or removal of elements that increase productivity or efficiency.
- Design so that the machine may operate at several locations along a production line to performing different tasks using the same basic structure.
- Should be designed using a modularity approach for common hardware and interfaces.

2.3 Supervisory Control and Data Acquisition

Supervisory Control and Data Acquisition (SCADA) systems are widely used to monitor and control operations in various industries, networks and processes. These include: electrical power distribution, oil and gas pipelines, water distribution, sewage treatment plants, manufacturing and the list continues. SCADA systems are intended to facilitate the work of a centrally located operator in charge of a widely distributed process by monitoring and gathering measurement information; locating, identifying and reporting faults; sending control commands and changing set points in distant controllers [22-25]. Based on the process information, status and alarms, the operator can make the necessary decisions to control the system equipment and keep a process on the right heading. It eliminates the need for technical staff to be present at, or travel to remote sites if the system is operating normally. It is evident that SCADA improves control capability over processes and conveniently saves time and effort by eliminating unnecessary travel and maintenance checks to remote sites.

2.3.1 SCADA components architecture

Essentially, SCADA systems consist of similar selections of components. Firstly, process controllers which can be either supervisory or device level (remote station) controllers. Supervisory controllers can be in the form of a Programmable Logic

Controller (PLC) or a networked computer. In contrast, device level or a remote station controller can be equipment like remote terminal units (RTU), intelligent electronic devices (IED) or PLCs – connected to the system sensors and actuators. Secondly, an HMI represents the window into a system. It is used to monitor the status of processes and enables an operator to input controlling commands into the system. An HMI can be a touch screen or panel, or a panel computer. In some cases, it can be a normal monitor with alternative methods of data input. In addition, SCADA systems also comprise database servers (historians). Its usage is to store measured system data and retrieve it again when reports are required. Furthermore, all SCADA components are integrated via a communication network or fieldbus [23-26]. A typical example would be Ethernet. To add reliability, a redundant network can be used in parallel with the main network in case of network failure. Lastly, a software platform on which the SCADA runs must be implemented. It must consist of functions to easily monitor and control the processes of the system. These functions will be discussed subsequently. Figure 2.2 shows the layout of a typical SCADA system.

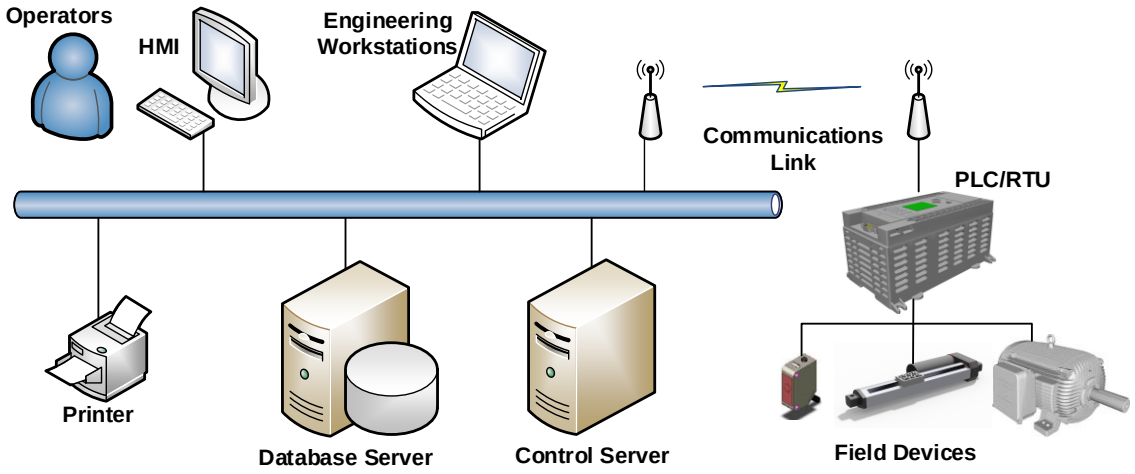


Figure 2.2 Typical SCADA system

2.3.2 SCADA functions

All SCADA software generally offers similar essential functionality. The capability to monitor and control a process by means of a graphical interface – which allows a user to view information and control the real-life process. In addition, SCADA also comprise the following functionalities: data acquisition and processing, alarms and

events handling, logging and archiving, trending and analysis, and historical data report generation [24, 26-28]. These are summarized as follows:

- ***Control:*** Users who typically have read/write access privileges to a system can access a subsection at a selected location and immediately issue commands (or control) to this subsection of a system. Users are usually allocated to different groups, which each has explicit control access to various sections of the system (see subsequently).

- ***Human Machine Interfaces:*** HMI screens are the windows into a system. These can support multiple screens at a time, which consist of various system diagrams and text. These screens typically are populated with graphical objects to represent process variables and display the status, min/max ranges, alarms etc. HMI screens also facilitate typical window editing like zooming, re-sizing, scrolling and navigating.

- ***Data Acquisition and Processing:*** Data acquisition is the collection of data and presenting it as process data back to the user. Data is acquired by scanning sensors and additional variable conditions during the system runtime processes. Data is then processed to display a conversion or manipulation of the data back to the user. This manipulated data includes data quality checks, analogue processing, limit checks, integrity checks, zero suppressions and calculated derived variables.

- ***Trending:*** Trending is an extension function on data acquisition. It plots process measurements on a selected scale to give information in the form of an ongoing graph. A trend typically shows the history of process data for a specified period of time (one minute, one hour etc.). It displays the progression of the data over a period of time.

- ***Alarms and Event Handling:*** Alarms typically alert a user to unplanned undesirable operating conditions. These alarms occur based on limit and status checks performed in the data acquisition section. The alarms are normally arranged by multi-level priorities, based on the severity and criticality of these alarms. User-defined alarms can be developed using arithmetic or logical expressions to alert users based on derived conditions.

Occurring alarms are usually handled centrally, where the true source of information exists. This ensures that all users see the same status of acknowledged alarms.

- ***Logging and Archiving:*** Logging and archiving can be understood as the ability to store process data. Logging can be seen as short- to medium-term storage of data, whereas archiving is long-term storage of data on disk. Logging typically stores data on a cyclic basis or when initiated by a data value change. Eventually, logged data can be transferred to an archive after an expired period of time or once a log is full. Logged and archived data can again be retrieved and viewed in the future.

- ***Historical Reporting:*** Historical reporting is simply retrieved logged and archived data that are formatted for reporting purposes. This retrieved historical data can be filtered and manipulated to show graphs, charts and figures. Afterwards, this data can be compiled into a report, then either be printed or saved as Excel, Word or PDF documents.

2.3.3 Access Security

As seen from the previous section, a SCADA system must implement access control as a security measure to avoid unauthorized access to the system. A user can gain access to the system by choosing a level of privilege, then either entering a user name and password or presenting a user ID tag (RFID). A user must be granted permissions to access the system with a respective level of privilege. These access privileges are developed into four different levels, namely [29, 30]:

- ***Level 1:*** Monitor access – a user can view processes on the screen, retrieve and view historical and real-time trends.
- ***Level 2:*** Operator access – a user has monitor access with additional privilege to change the state of equipment (on/off), and change the set points for control values.
- ***Level 3:*** Supervisor/engineer access – a user has operator access with additional rights to change process control parameters and system alarm settings.

- ***Level 4:*** Administrator access – a user has supervisor access and additional maintenance privileges. Here the user can modify settings in the system operating software.

By default, the system will operate in access level 1, unless a user logs in with a different privilege level. During this time, the current privilege level and the user name should display in the alarms and events window at the bottom of the screen. After a period of inactivity, the system will automatically return to the default access level.

2.3.4 Generations of SCADA

There are four generations of SCADA; these are declared as follows:

2.3.4.1 First Generation: Monolithic SCADA systems

For first-generation SCADA systems, computing was done by minicomputers. Ordinary network systems were not yet available at the time, thus SCADA had no connectivity to other systems and used strictly propriety communication protocols. The architecture consisted of a single mainframe system connected to remote sites via RTU [31, 32].

2.3.4.2 Second Generation: Distributed SCADA systems

In second-generation SCADA, information and control functions were shared across multiple operator stations connected through a Local Area Network (LAN), hence a distributed system. Individual stations were responsible for performing particular tasks. This reduced the overall size and cost compared to first generation systems. However, network protocols were not yet standardized, thus propriety protocols were used. This resulted in a situation that security of the SCADA installations were generally ignored [31, 32].

2.3.4.3 Third Generation: Networked SCADA systems

Networked SCADA systems are generally connected through a Wide Area Network (WAN) system, also known as a Process Control Network (PCN); and communicates using either Ethernet or fibre optic connections for data transmission between SCADA nodes. In addition, several parallel working distributed SCADA systems operate under a single supervisory controller in the network architecture.

This generation of SCADA uses PLCs at remote nodes for monitoring and control; and occasionally flags an operator in cases of major decision requirements [31, 32].

2.3.4.4 Fourth Generation: Internet of things technology

With the availability of cloud computing, fourth generation SCADA systems have adopted the “Internet of Things” technology, also known as “Industrial Internet of Things”. It significantly reduces infrastructure costs and increase ease of maintenance and integration compared to the earlier SCADA systems. As a result, SCADA systems can report using cloud environments and implement complex control algorithms. Furthermore, the use of open network protocols provides a more comprehensible and manageable security boundary than that of mixed proprietary network protocols used in decentralized SCADA systems [33, 34].

2.4 Advanced Manufacturing Planning and Scheduling

Manufacturing companies face the challenge of increasing competition for global markets and have to manage all functions of the company impeccably. To accomplish this, manufacturing companies must progress towards adopting advanced systems for planning and management. Solutions for planning and scheduling systems include, but are not limited to, enterprise resource planning (ERP), advanced planning and scheduling (APS) and supply chain management software systems [35-39]. Traditionally, manufacturing companies would turn to ERP systems. These have evolved over the years from the initial material requirements planning (MRP) systems to manufacturing resource planning (MRP II) [40, 41] to the early versions of ERP systems. ERP systems allow seamless integration between application programs used across all the different functions/departments (human resources, finance, sales, marketing, development, production planning) of the company. In addition, an ERP system is largely utilized for its planning capabilities, as well as its abilities to support decisions regarding the planning and execution (managing) of the business. On the contrary, current ERP systems are somewhat lacking in decision support and planning in a dynamic environment. This is, however, resolved by integrating an ERP system with supporting APS systems and manufacturing execution systems (MES).

APS systems are flexible strategic planning software/modules that utilize information like labour, material and equipment to determine the best supply chain schedule with multiple given constraints. APS systems allow for real-time adjustments in case of unplanned events and while ensuring optimal supply chain throughput. Together as a complete integrated planning and scheduling system, these perform the managerial process by which materials (parts in inventory), resources and production processes are optimally assigned to achieve a desired manufacturing demand before an allotted time. The integrated system considers due dates, availability of required raw materials to complete products; cycle time of manufacturing processes (sequences); and the required resources (staff and equipment) to complete processes to develop a reasonable and attainable schedule. The result is a fully integrated facility-wide solution to plan, manage and execute manufacturing processes and, at the same time increase productivity, reduce operational costs and provide the ability to promise.

2.5 Graphical Design for Human Machine Interface Screens

2.5.1 Introduction

HMIs, also known as Man Machine Interfaces (MMIs), can be either an assortment of LED indicators and mechanical switches, capacitive touch panels, or industrial control panels with LCD displays that are used to monitor and interact with SCADA systems. This section concentrates on the proper and effective usage of graphics to design the interface screens for these HMIs.

HMI screens are the primary method to relay important operational information to a user. These HMI screens has to be designed very efficiently to relay the maximum amount of information to users, without overwhelming them. How clear and effective these displays are designed, determine how well a facility is operated. In addition, the design of effective HMI screens plays a critical role in a user's ability to effectively manage a process or operation, especially in the case of abnormal situations. With this in mind, two major factors should be taken into consideration when designing HMI screens; the screen must be able to hold the attention of a user with maximum display clarity; and the design must allow users with little or no training to be able to successfully operate a system intuitively [42].

2.5.2 Considerations for Graphical User Interface Design

There are important considerations regarding the design of effective high performance HMI screens that determine how HMI objects are displayed and positioned on the screen. These considerations include the layout of the screen; the use of colour for text, objects and backgrounds; the way process information and status are depicted; as well as the use of alarms and animations. These considerations are discussed subsequently.

2.5.2.1 Screen Layout

A major aspect in designing effective graphics for high performance HMI screens, is the layout of a screen. The layout of controls, indicators and other important information on the screen must be arranged in such a way as to make sense logically and keep screen clutter to a minimum (what belongs together, must be placed together). Failure to avoid screen clutter will result in data getting lost on the screen. Before HMI screens can be designed, it is useful to identify how a user will perceive it. Typically, a user will scan an HMI screen in a similar manner as a page from a book [42, 43]. This means starting at the top left corner, proceeding to the right and then continuing down the screen. Since an HMI screen has no lines to guide the user's eyes, a user will generally perform a few incomplete scans of the screen. With this in mind, advantage should be taken of placing important information and objects in the areas within the screen where attention is easily drawn to. It is therefore recommended that the summary of alarms should be placed across the top of the screen [42, 43]. Furthermore, any graphical images should be placed to the centre and left of the screen, with supporting key information to the centre and right of the screen. It is recommended that control and navigation buttons, along with any supporting graphics and company logos, be placed in the lower section of the screen.

2.5.2.2 Usage of Colour

Colour is a powerful tool to enhance the visual presentation of key information on the screen. In contrast, if colour is wrongly used, it can be misleading, overwhelming and even dangerous. This means that it is important to use colour smartly and carefully. The use of colour is discussed below.

- ***Usage of Colour to Display Screen Objects***

It is evident that bright vivid colours draw the attention of the human eye. With this in mind, bright colours must be reserved to be used for alarms and abnormal situations, not for regular conditions. For example, regular running status should not be displayed in vivid saturated colours, such as red or green (primary colours), but rather reserved for alarms and events. In addition, graphical objects containing large groupings of primary colours should be avoided, because these will cause complementary colour image retention on the retina – also known as after images [43, 44].

Essentially, it is important to predefine a colour convention to be used and use it consistently. Safety colours that are meant for alarm purposes should not be overused for other purposes such as status depiction and ordinary graphics. If a colour is wrongly and inconsistently used, it ceases to have impact and can lead to misinterpretation or confusion. The preferred colour convention used as a standard is well defined by literature and is as follows [45]:

- **Red:** Danger, Prohibition, Emergency
- **Yellow:** Warning, Caution, Risk of Danger
- **Green:** Safe Condition
- **Blue:** Mandatory or Compulsory Operation

- ***Background Colours***

By referring to the preceding section, primary colours should certainly not be used as background colours in an HMI screen (after images). On the contrary, blue is the only primary colour that can be used as a background and makes a good one (blue cones in the retina). Black and white deliver good colour contrast for text; however, black and white produce too much screen glare in bad lighting situations and are therefore strongly discouraged. In fact, it is recommended to rather use muted tones or pastel shades like light grey, light brown and blues as background colour. These colours provide great contrast for brighter colours used to display miscellaneous information as well as vivid colours used to display alarms and events. In addition to choosing the perfect background colour, using slight variations in shading can create an illusion of raised and lowered sections, which will make it easy to

distinguish between areas of a screen (grey scale). This will ensure that a user will immediately be able to identify between different sections of the HMI screen.

- ***Displaying Text***

Text is the most versatile method to convey important information to a user. Due to this fact, text should not be difficult to read because of type of font, size and colour contrast. To ensure the best possible contrast, text should generally be black in colour, unless there is a good reason to use another colour and the best possible contrast with background is achieved. Examples of good and bad contrast between the text and the background can be seen in Figure 2.3.

In addition to text colour, it is necessary to choose the right font to avoid misperceptions. It comes highly recommended to select a font that is common and exists on most computers. Examples of these include Arial, Times New Roman, Helvetica and Courier. This will ensure that text will display the same even if the HMI application is transferred to another computer. If unusual fonts were used in the HMI application, it might be found that the fonts are re-mapped to an unexpected gothic script, which is undesired and can be difficult to read. Due to the low resolution of most HMI screens, it is better to use a San-Serif font such as Arial, because the screen resolution might not be high enough to clearly render the detail of a Serif font. In addition, the size of text should be such that a user can read information on a screen from some distance away without any difficulty (from across the room, for instance). A good starting point is Arial at 16pt and up to two larger sizes for headings and labels. It should be avoided to have more than three different sizes of text in the application. Too much variation in text size might make the HMI screen seem cluttered and become confusing. Furthermore, the usage of upper-case letters and underlining should be limited and reserved for headings to prevent eye strain. Text should generally appear in lower-case with the first letter of the leading word capitalized. While this is recommended practice, common conventions include capitalization of each word. Whatever the convention chosen, it should be used consistently.

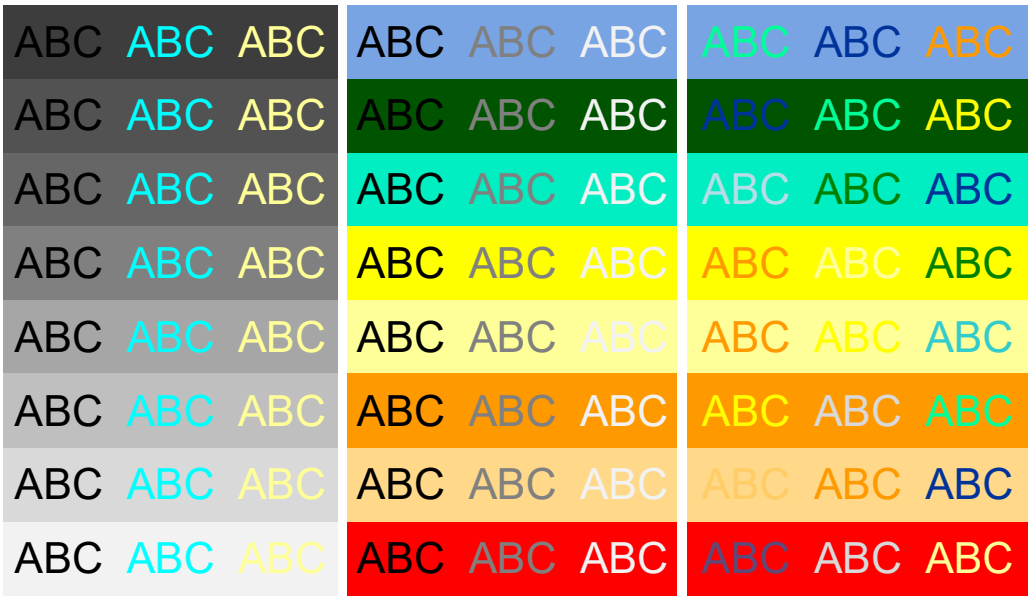


Figure 2.3 Example of text and background colour contrast [46]

2.5.2.3 Data and Status Depiction

What is displayed on an HMI screen; data or information? What is the difference between data or information? To ensure that HMI screens are most effective, the difference between data and information should be clarified. Information is data that is given context. For example, refer to Table 2.1 and cover the "Range" column. Scrutinize the data and determine if the person with these vitals are healthy. Unless explicitly trained as a medical professional, how can it be observed that this person is healthy or not? It should now be evident that data requires context.

Table 2.1 Data that is given context [47]

Blood Test	Results	Range
HCT	31.7%	24.0 - 45.0%
HGB	10.2 g/dl	8.0 - 15.0 g/dl
MCHC	32.2 g/dl	30.0 - 36.9 g/dl
WBC	9.2 x 10^9 /L	5.0 - 18.9 x 10^9 /L
GRANS	6.5 x 10^9 /L	2.5 - 12.5 x 10^9 /L
L/M	1.3 x 10^9 /L !	1.5 - 7.8 x 10^9 /L
PLT	310 x 10^9 /L	175 - 500 x 10^9 /L

- ***Presenting Data Values***

As seen from the previous subsection, the way information is displayed to a user is of high importance. Data should always be given context to become information. Data values that are randomly placed around a screen are often hard to perceive. In essence, values that belong together, should be grouped together. It is also best practice to place values that need to be compared in a table next to each other. If these values have multiple data types (temperature, pressure, speed), then the order in which these are displayed must be the same in each following table. In addition, if these data values have different units, then the units should be declared and properly labelled. On the contrary, if the units for these data types are apparent, then it is better to leave them out to avoid unnecessary screen clutter. Furthermore, it is preferable to display data in a graphical manner like an analogue meter, trend or graph. Humans understand and respond much quicker to data that is presented graphically than numerically [48]. For example, the process to compare each data reading to a memorized translation of what a good value should be can be a demanding cognitive procedure. As the number of values displayed on the screen increases, then the process of translating this mental map becomes slower and rather distracts a user. However, if these values were represented as a group of analogue indicators (see Figure 2.4), then a user will intuitively understand the value depictions because humans are internally hard-wired for pattern recognition. At a single glance, a user will be able to tell whether a reading is outside the normal ranges, by how much and how far from an alarm occurring. Thus, in a series of short scans, the user will be fully aware of the overall system performance. Additionally, if a user requires to know the exact value of a reading, then it is recommended that a numerical value be presented along with the analogue depiction. In this case, the appropriate data resolution should be used and unnecessary decimal places should be avoided.

Another method typically used to display levels in a tank/container involves vessels. Referring to Figure 2.5, it comes recommended that vessel levels should never be displayed as large blobs of saturated colour. This can cause after images and also wastes unnecessary space on the screen. A simple bar or strip depiction that indicates the desired operating ranges represents a better utilization of screen space. An even better usage of screen space would be vessels that display a

combination of trend data and analogue indicator depictions (See Figure 2.5). The latter capture the knowledge of what is normally embedded into the vessel display. It clearly shows the operating ranges, the current value, where alarms would occur and a period of operating history (trend), which is highly desirable.

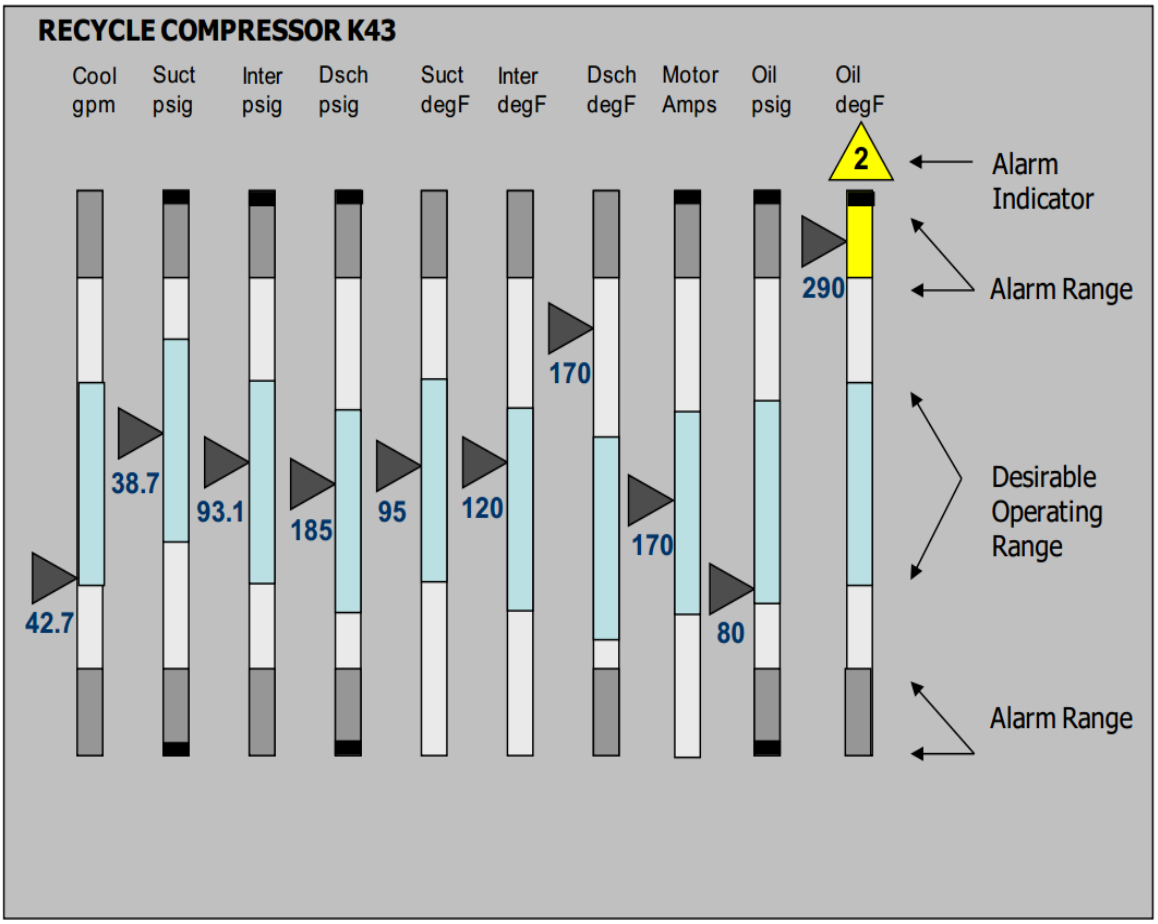


Figure 2.4 Analogue depiction of information [49, 50]

- ***Status Depiction and Animation***

Recalling from Section 2.5.2.2, colour conventions should be maintained consistently. This gives rise to common paradigm that bright green depicts “on” and bright red depicts “off”, or vice versa, depending on the industry. The reasoning behind these conventions is that green is to indicate a running plant process and red for stopped. On the contrary, alarm logic would suggest that red is dangerous (running) and green is safe (stopped). To avoid confusion, it is a better practice to adopt a convention that does not use red and green. Figure 2.6 shows the use of a colour brighter than the background for on, and a colour darker than the background for off. Imagine that the displayed object has a light source inside it – dark equals

off and light equals on. Additionally, it is recommended to add redundant coding to display the status as descriptive text next to the object.

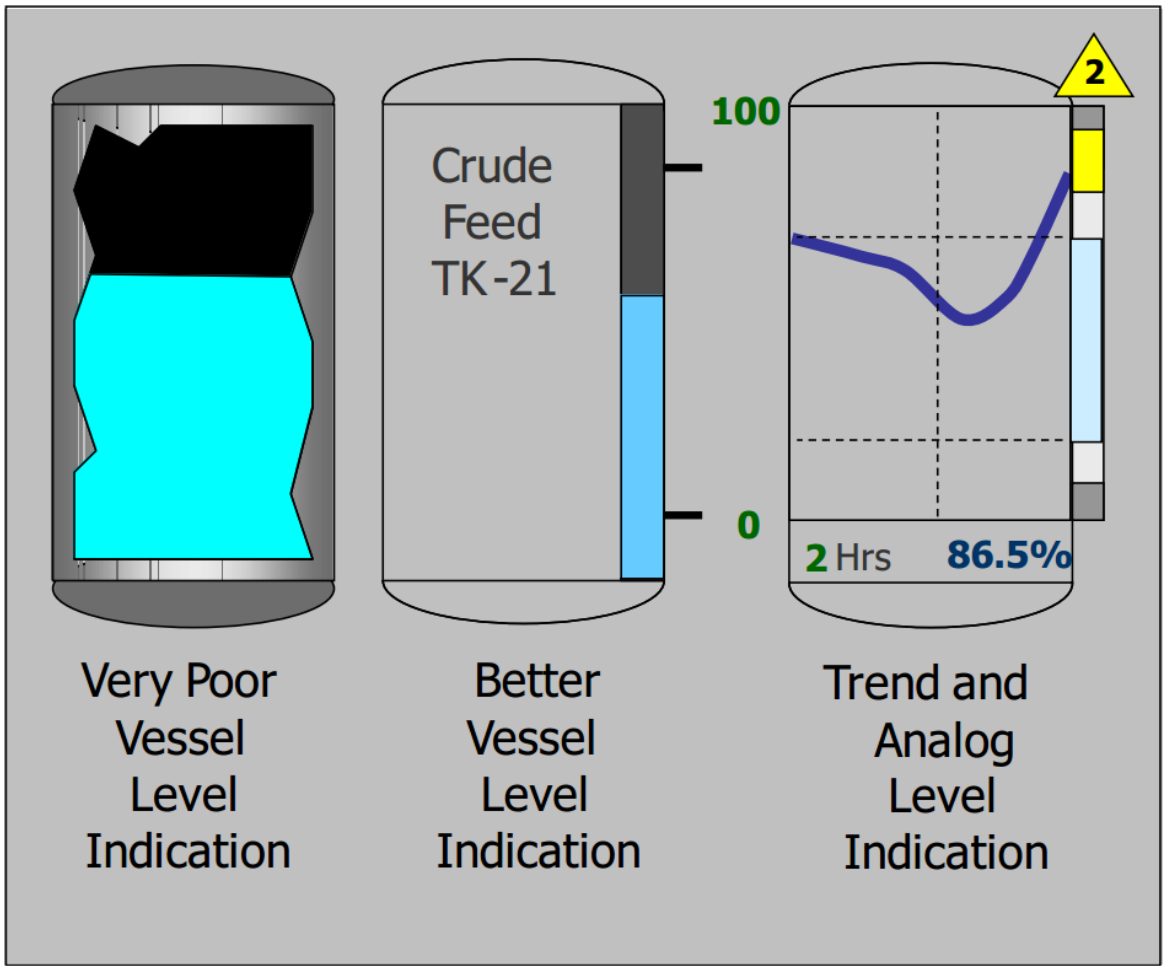


Figure 2.5 Vessel levels [49, 50]

Process status is frequently depicted by colourful animations of physical equipment. In fact, HMI graphics should not contain any bright-coloured, three-dimensional drawings of equipment, nor spectacular animations of flashing lights, spinning pumps and fans, or moving conveyors. In general, exact representations of a facility should be avoided as a rule. In some cases, a well-done depiction of a system can often beneficially aid a user in immediately visualising the facility and the location of the measurements (used for detailed views), but using too much detail can clutter a screen and cause important information such as alarms to be unnoticed. It is a good idea to design HMI graphics in shades of grey (grey scale). Adding indicators for the 3D equipment can make the display more identifiable, but redundant 3D images of pipes, pumps, valves, etc. should be avoided to keep the display uncluttered. As a rule, realism is best avoided.

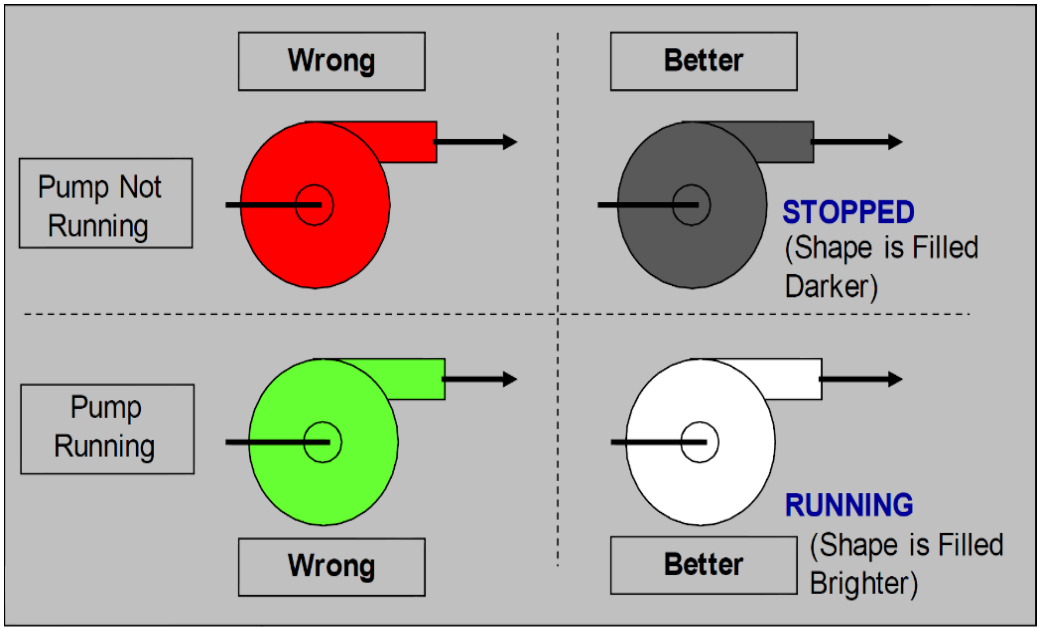


Figure 2.6 Status depiction with redundant coding [49, 50]

2.5.2.4 Navigation

It is typical in multi-screen HMI applications that a user requires to navigate between system screens. This must happen quickly, easily and intuitively. HMI designers typically use either of two approaches to achieve this; navigation buttons or a picture with embedded “hot spots.” For the first approach, navigation buttons are used that should be clearly labelled and large enough to be pressed regardless of the pointing device used in the application (fingers or mouse-cursor). These are typically placed near the bottom section of the screen. In addition, these navigation buttons should be grouped together and placed at the same location on each screen. This will limit traveling hands and make it easy to locate the buttons. The second approach used by HMI designers is to embed a “hot spot” in a schematic representation of the entire system layout. A user can quickly and easily navigate to a detailed subsection view, by simply clicking on the desired area where the subsection is displayed in the system overview. Furthermore, a well-designed HMI should have a return to overview or main button on each screen, to navigate back to the top level screen of the application (similar to a “Home” button used in websites).

2.5.3 Alarms and Events

Alarms and events should preferably be organized into a banner or summary, be visible on each HMI screen and preferably located at the top of the screen. It must be easy to navigate to a detailed screen with additional information about the

alarms. In addition, the colour conventions for alarms and events must be well defined and should not be used for any other purposes. Furthermore, it is important to mention that 1 in 12 men have some degree of colour-blindness [51], which will affect their perception of red and green. With this in mind, colour should not be the only differentiator when it comes to alarm depictions. To clarify, it is good practice that all alarms should also be redundantly coded to display a pictorial change, text, animations, or even have audible tones. Proper alarm depictions are redundantly coded to display in different colours with a supplementary pictorial change and descriptive accompanying text that are based upon the priority of alarms (different colour, shape and text for each priority) (see Figure 2.7). Additionally, it is a good idea to display alarms with separate alarm indicators next to the objects that are in alarm. These indicators should flash while unacknowledged and cease to do so after acknowledgement, but still remain while the alarm condition is true. This is the only proper usage of animations in HMI screens. Audible tones can be used to intensify alarms, but must provide a user the option to silence it by acknowledging the alarm. Whichever convention is used for alarm depictions; alarms should be located where it can easily be seen and accessed to be serviced/acknowledged.

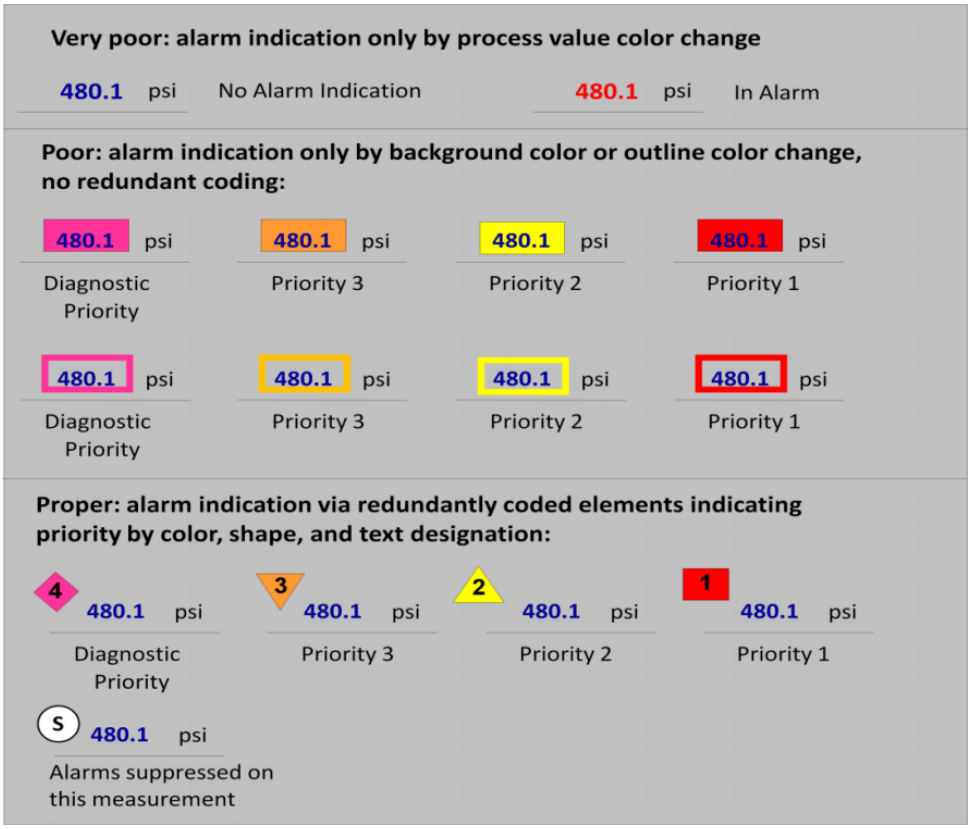


Figure 2.7 Depiction of alarms [49, 50]

2.5.4 Display Screen Hierarchy

Since SCADA systems comprise a multiple number of HMI screens, importance should be placed on the optimal hierarchical arrangement of it. Top level screens must be an overview of a system which displays overall process operations at a given time, while more detailed information must be revealed as an operator moves deeper into the display hierarchy in a logical manner. Referring to literature, it is desirable that a system display hierarchy consists of four levels, and each level can be distinguished as follows [50, 52, 53]:

2.5.4.1 Level 1 – Operation Overview

A level one screen gives, at a single glance, a complete overview of the operations and status of the entire system at a facility. This screen provides a clear indication of all the current processes and is essentially a summary of all the key performance indicators of all subsystems combined onto one screen (see Figure 2.8). Moreover, this screen is intended to be used as an overhead screen, but is, however, not used to perform control interactions. Furthermore, important information that is typically displayed on this screen will include: top priority alarms and events, major equipment status, important trends and process parameters, along with other key performance indicators.

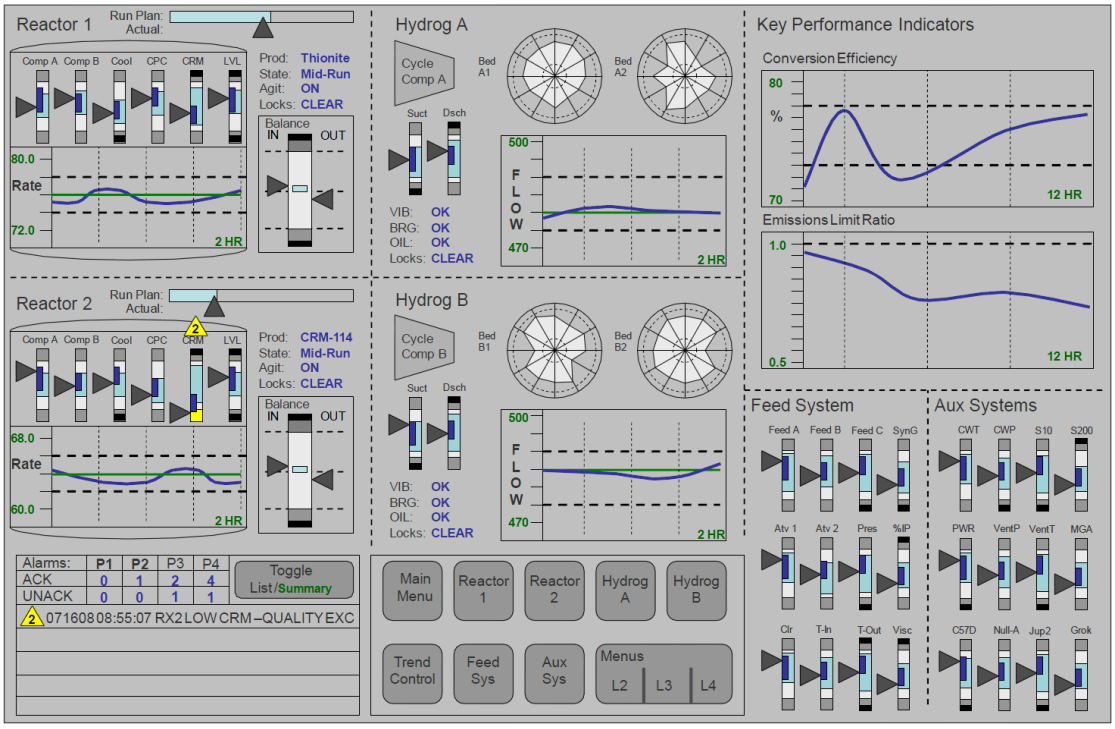


Figure 2.8 Example of a level 1 screen [50, 52]

2.5.4.2 Level 2 – Unit Control

An entire system can always be divided into smaller sectional operations and subsystems. Due to this, a level two screen exists for each unique subsystem present in the system. This level screen shows all the information and controls required to perform most of the operational tasks related to the subsystem in focus on the screen. These screens are primarily for monitoring and control, and are where operators will perform the majority of device tasks. In addition, these screens provide an overview of controllers, actuators, trends, alarms and the status of each subsystem.

2.5.4.3 Level 3 – Unit Detail

A level three screen provides a more detailed view of a single device or piece of equipment compared to a level two screen. These typically have a schematic type layout and are utilized to perform diagnostic operations and troubleshooting. In addition, the items displayed on a level three screen may typically include: all instruments, interlock status, troubleshooting displays, control loops and schematics.

2.5.4.4 Level 4 – Support and Diagnostic Displays

Level four screens present the most possible detail of a subsystem and subsystem components. These screens will display individual sensors and actuators, subcomponents and detailed diagnostic information. In complex systems, a level four screen is normally utilized to support documents and information. Moreover, these documents and information will include operating procedures, alarm documentation and guidance.

2.6 LabVIEW and DSC Module

- ***LabVIEW***

LabVIEW, which stands for Laboratory Virtual Instrument Engineering Workbench, is National Instruments' graphical programming (G programming) development environment. LabVIEW is a powerful tool used in various industries to integrate software with a wide range of hardware, in order to perform tasks like data acquisition and processing, analysis, logical operations, process control, report generation, as well as other areas of research and development. G programming

functions on a “dataflow” principle where an output value can only be obtained once all the inputs connected to it are provided with values. The code is developed in what is called virtual instruments (VIs), and can be seen as modular programs or subroutines. These VIs each consists of a front panel (Figure 2.9 at top) and a block diagram (Figure 2.9 at bottom). The front panel is essentially the graphical user interface (GUI) of a VI, and consists of controls (switches, knobs, numeric inputs, etc.) and indicators (LEDs, graphs, numeric outputs, etc.). These controls and indicators each obtain its behaviour based on how the functions on the block diagram is wired (connected). The block diagram which is the background code contains the functions and variables (Boolean and arithmetic functions, programming loops and structures, etc.), and also specify how these are interconnected to realise the required behaviour on the front panel [54-56].

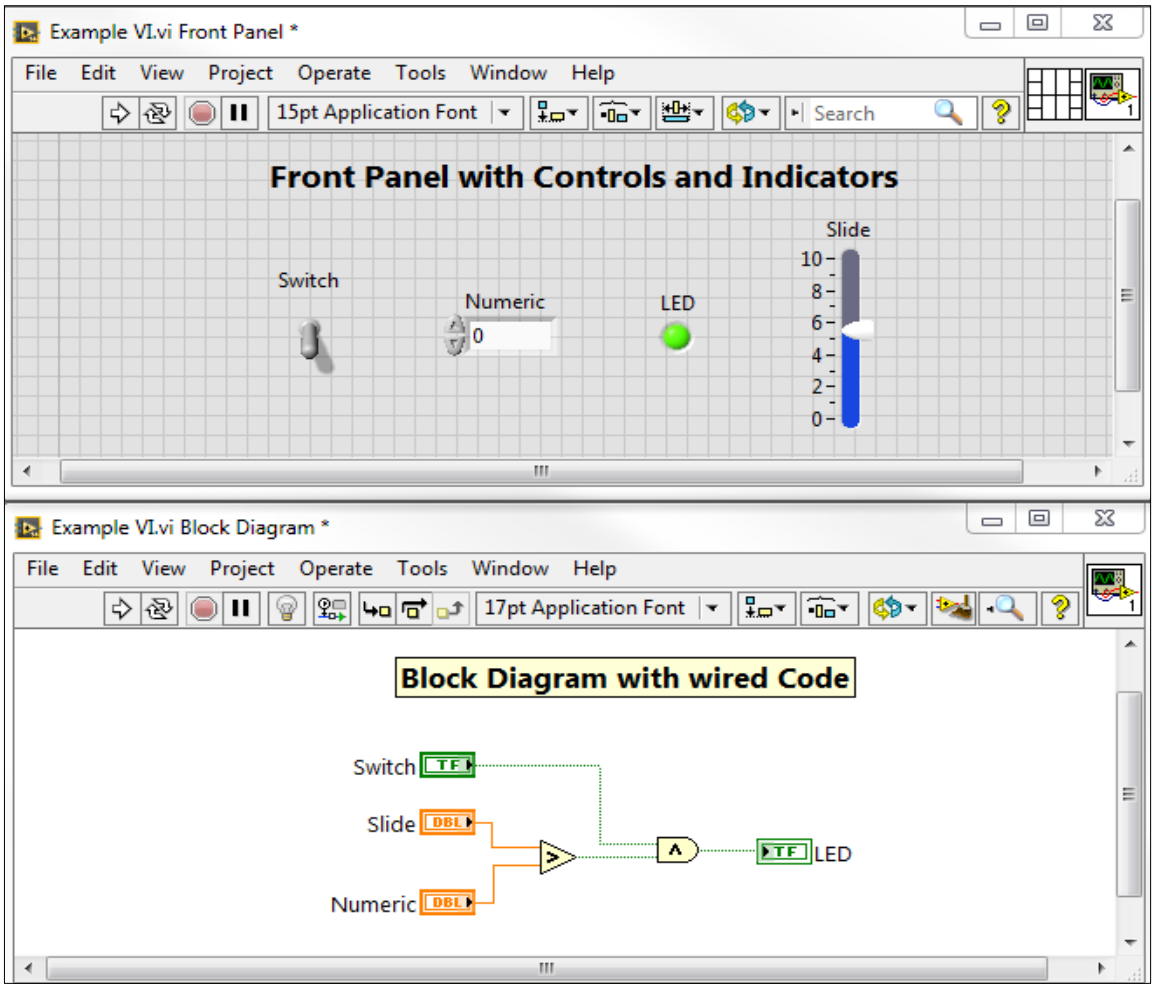


Figure 2.9 LabVIEW front panel and block diagram

- ***NI DSC Module***

LabVIEW comprises multiple native tool pallets as well as additional industry specific tool pallets. The Data-logging and Supervisory Control (DSC) module is a workbench (tool pallet) that are specific to the development of SCADA applications. The DSC module utilizes a shared variable engine (SVE) to host and manage network published shared variables (SV), through the use of proprietary technology called NI Publish-Subscribe Protocol (NI-PSP). This is accomplished by creating libraries, also known as processes, in which the SVs resides. These libraries are then deployed to the SVE to host the SVs. In addition, these libraries can be utilized to create OPC (OLE for Process Control) bound variables, which enables the system to communicate with a wide variety of software and hardware devices. Furthermore, the DSC module provides functions set up and control SCADA features like alarms, events and logging. All considered, the DSC module delivers a complete package to successfully develop and implement SCADA applications [57-59].

Chapter 3 Methodology

3.1 Introduction

The research problem at hand clearly indicate that a reconfigurable assembly system, and its monitoring and control, need to be extremely flexible to compete for global markets. This chapter discloses the methodologies undertaken to develop such a system, along with a system to monitor and control it. These methodologies include the following: identifying the architecture of the entire system (that would ultimately be tested); selecting the hardware to construct the system and to assemble the physical system devices to be used (build up SMART conveyors); a means of interconnecting system devices; specifying the software architecture of the system and the development of the required software modules. All considered, this chapter provides the methods to develop the system under study in entirety.

3.2 System (HCoMS) Architecture Overview

The characteristics of reconfigurable assembly systems were discussed in Chapter 2, where it is evident that these systems must be modular in design. Since any compliant device can be connected to the HCoMS controller, only some devices will be handled as part of a case study. To clarify, the devices used in this study were chosen for explanation purposes, but can be replaced by any other device which meets the requirements to be compatible with the HCoMS controller (has an HCoMS compliant device installer developed for it, and are modular in design).

The system architecture of how the physical system components interconnect can be seen in Figure 3.1. Firstly, the system uses OPC as a common communication protocol via Ethernet to resolve incompatibility issues between various devices from different vendors. At the heart of the system is the main controller with the HCoMS main application running on it. The main controller is a high specifications panel computer with LabVIEW, an OPC server, as well as a database server installed on it. In addition, all of the system components are connected to the main controller through an Ethernet network. Among the required devices needed to assemble products are industrial robots, machine vision stations, and modular conveyor cells called SMART conveyors. These devices are all similar in structure. Each has an OPC complaint PLC with intelligence, which is wired to sensors and actuators, to

control the operations of the device. This means that each device has intelligence and the main controller only supervises and instructs what each device must perform. Furthermore, users can control, monitor and interact with the system by making use of monitors, peripherals and touch screens linked to the main controller. The system provides users with the option of interfacing with the system through tablets and smart phones.

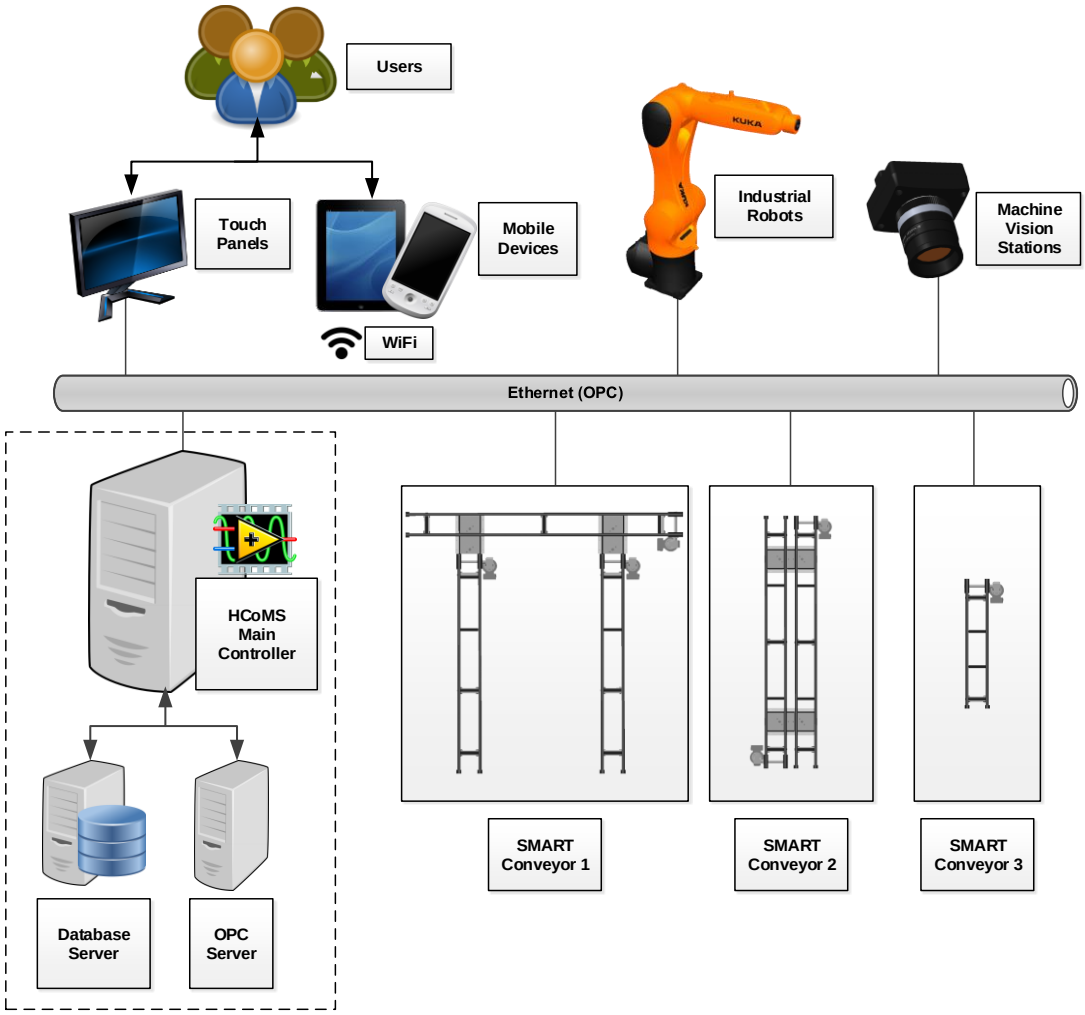


Figure 3.1 HCoMS architecture

3.3 System Hardware Components

The subsequent sections deal with the system hardware components that are chosen to be utilized in an HCoMS. All of the components used require a device installers developed for each in order to operate within the HCoMS, with the exception of the OPC server. In addition, the subsequent sections also discuss the physical structure along with the functional behaviour of each device used in the system.

3.3.1 OPC and NI OPC Server

3.3.1.1 OPC

OPC is a communication platform widely used in industry to interface between a variety of incompatible hardware and software devices. In addition, OPC is a standards specification which compels that devices must transfer data in a standardized and usable format. Moreover, each application must implement at least one OPC compliant driver to access data from an OPC server. These OPC servers provide the platform for software clients to access and control production data from process control devices (like PACs and PLCs), provided that these devices are OPC compliant and are connected to the same Ethernet network [60].

3.3.1.2 NI OPC Server Setup

LabVIEW offers a solution, namely NI OPC server. NI OPC server is natively installed with National Instruments' DSC (data logging and supervisory control) module and is also widely known in industry as KEPserver from KEPware, which have support drivers for numerous hardware devices from multiple vendors. The setup for NI OPC server will now be discussed briefly.

Firstly, an NI OPC server must either be installed on the same computer as the main control application or on a separate server computer, as long as it is networked with all the system devices. At this point, a new server configuration can be configured; a previously saved configuration can be loaded; or an existing server configuration can be altered to suit the system requirements. However, the core intention of an HCoMS is that the system setup, changeover and calibration must be done easily and intuitively with minimal effort in a minimal amount of time. To explain the server setup thoroughly, a new server configuration will be dealt with.

After the initial setup, a new server configuration can be opened, and a new communication channel can be added and renamed as the device to be set up. The channel is then set up by choosing the type of communication (Ethernet, serial, etc.) and which network adapter in the computer will be used. Next, a device can be specified under the newly added channel, by selecting the model of the device to be used, and by specifying the device IP address. After the channel and device are fully defined, the OPC tags can either each be manually created, or simply imported

from a CSV file. Informatively, the manual creation of tags entail that the tags are each individually added to a device, given a user-definable name, specifying each tag address and modifying the tag properties. This is thoroughly explained by the author in [60].

On the contrary, an easier route can be taken. The user can right-click on the desired device and select import CSV (Figure 3.2). The user will be prompted to select a file to import from. Here the user can choose the CSV file included with the driver for the device. After the import is complete, the device will be populated with the tags specified in the device driver (Refer to Figure 3.3). These easy steps need to be done only once for every device that gets added to the system. Once all the required devices are configured, the server setup is complete and can be used. An example of this setup procedure is provided as a support video that is attached in Appendix A.

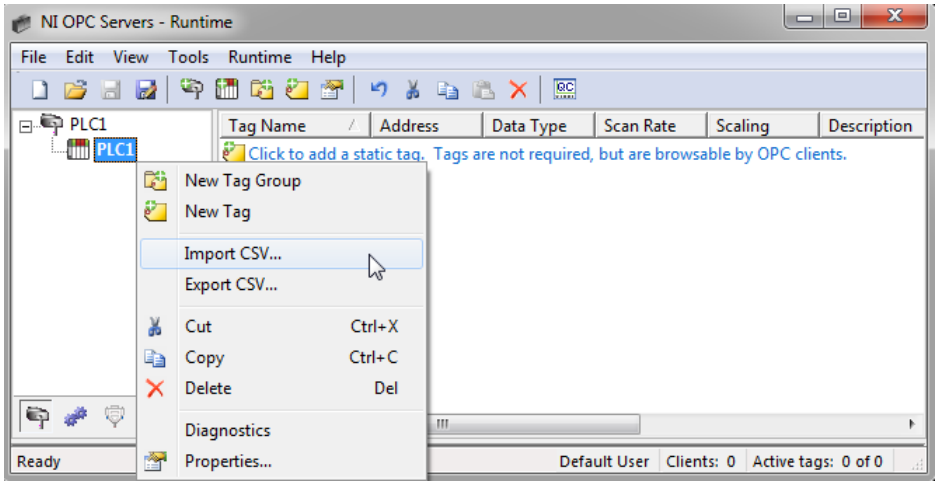


Figure 3.2 Importing OPC tags from .CSV file

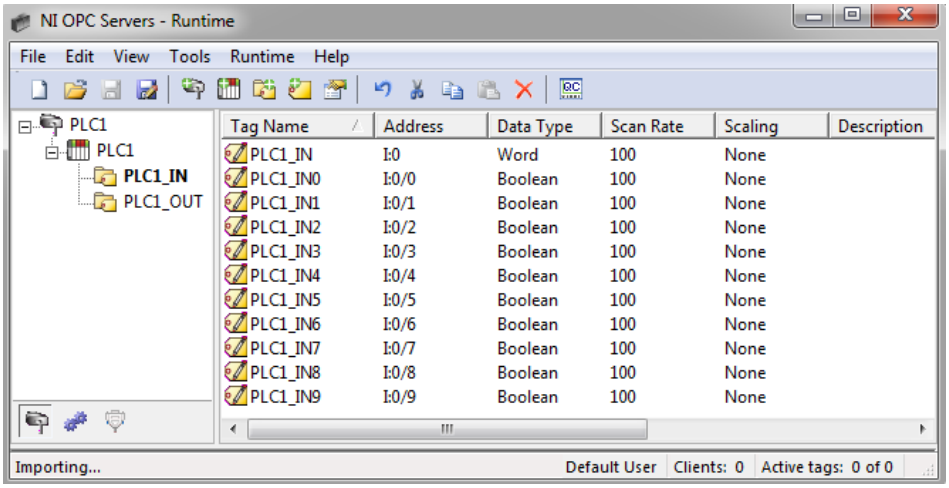


Figure 3.3 Completed server setup

3.3.2 SMART Conveyors

Products as well as the parts needed to assemble products are transported within the system using the SMART conveyors. The SMART conveyors are fundamentally combined sections of modular conveyors that are wired to controllers (PLC), sensors and actuators (stop gates and motors) which collectively form the construction of a cell of conveyors with intelligence. The modular conveyor sections used in the SMART conveyors are the TS1 system from Rexroth and are selected based on a prior study done within the RGEMS research group [60]. The Rexroth conveyors are chosen mainly for the modularity of it, which makes it possible to easily expand, change or rearrange the conveyor sections. An example of the different SMART conveyors that are used in the system are depicted in Figure 3.4.

All SMART conveyors are related in structure. As mentioned in section 3.2, each has an OPC compliant PLC with programmed functionality inside which connects to the HCoMS Ethernet network. However, each SMART conveyor differs in the number and size of the conveyor sections; the arrangement direction and orientation of these; as well as the number and location of the sensors and actuators that each has.

Figure 3.4 SMART conveyor systems

In addition to physical structure, the SMART conveyors are also comparable in terms of software structure. As a result of variations in physical structure, each

SMART conveyor has different distinctive capabilities that it can perform. An example of the different capabilities that one of the SMART conveyors can perform are shown by Figure 3.5. A SMART conveyor adapts the capability of it based on the information provided from the HCoMS controller. Figure 3.5 (at 1) shows an example of what the SMART conveyor executes with a particular program number provided to it. In addition, it can clearly be seen that the SMART conveyor changes behaviour as the program number changes (see Figure 3.5 at 2 and 3).

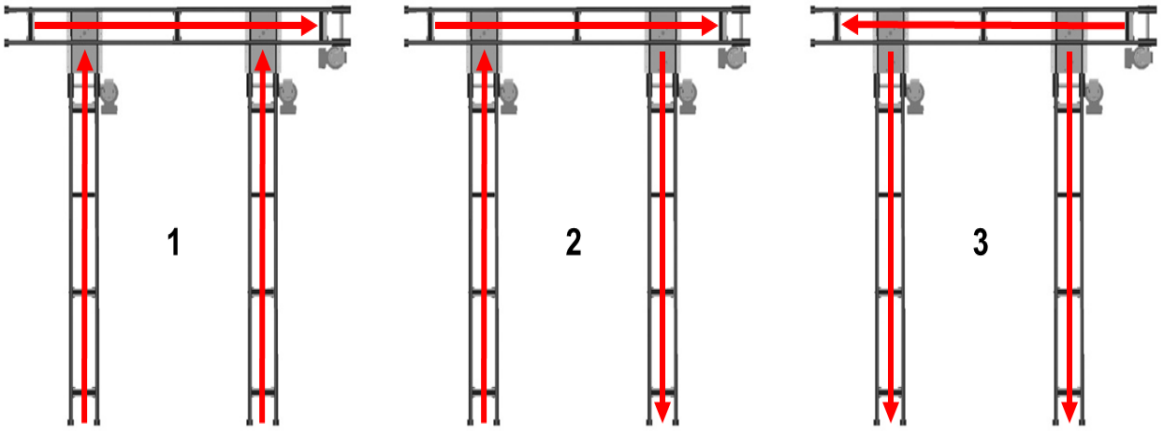


Figure 3.5 Change in SMART conveyor ability by change in program

Furthermore, the operation of the SMART conveyor software can be better explained by referring to Figure 3.6, which shows the general program structure of each SMART conveyor. Once a SMART conveyor is powered and connected to the HCoMS network, the PLC initializes and establishes communication with the HCoMS controller, and acknowledges that it is ready to operate. Here the SMART conveyor remains idle until the HCoMS controller issues a program number to it. After the PLC receives a program number, it adjusts the behaviour accordingly. Additionally, the HCoMS controller directly accesses sensor data from the SMART conveyor via OPC to control the overall process. The SMART conveyor PLC will continually execute the current behaviour until the HCoMS controller provides it with a different program number.

As a result, SMART conveyors are modular standalone devices with intelligence that connect to the HCoMS main controller that performs the supervisory monitoring and control.

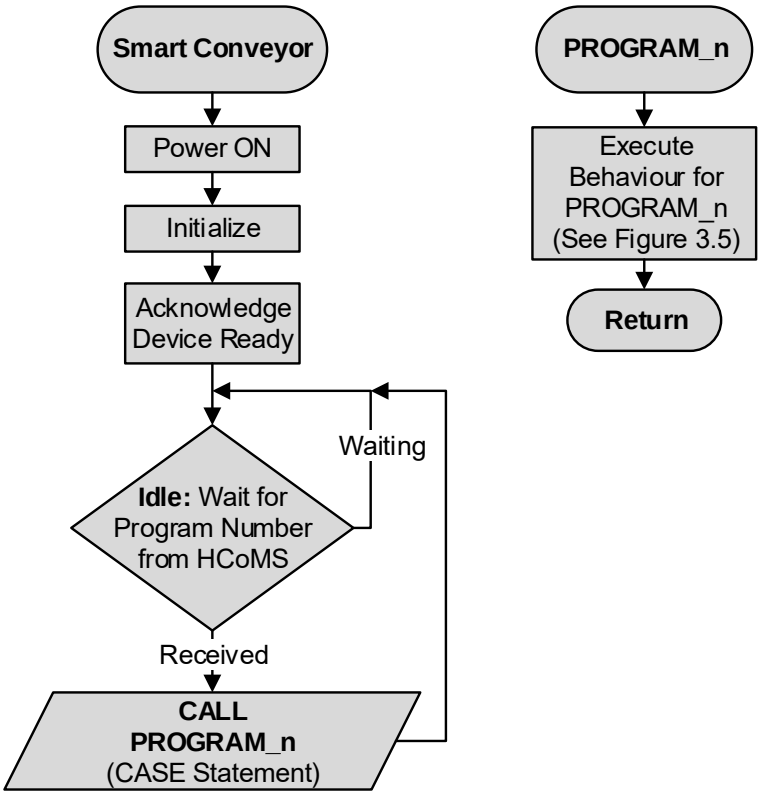


Figure 3.6 PLC program for SMART conveyors

3.3.3 Assembly Devices (Industrial Robots)

In addition to SMART conveyors that transport parts and products through the system, the function of the assembly devices depends on what it is needed for. In this case, the robots are either used to pick and place parts to assemble products or to transfer product pallets between different locations on conveyors. Figure 3.7 shows the KUKA industrial robots that are used as assembly devices in the system. These are chosen based on the modularity, speed, flexibility and agility, and high accuracy in repeating tasks that the robots can perform, which is needed for reconfigurable assembly systems, as explained earlier.

Like the SMART conveyors, the IOs of the KUKA robots are directly wired to an OPC complaint PLC, which connects to the HCoMS controller via Ethernet. The PLC is programmed to directly control the KUKA by providing it with instructions and receiving feedback from the surrounding sensors. It is worth mentioning that an OPC server can be directly installed on the KUKA robot, which would mean that the intelligence of the assembly device would reside inside the robot. However, this will require additional cost, licensing and extra protocols; and will not be included in this project.

The operation of a KUKA robot can be explained by referring to Figure 3.8. Firstly, once the KUKA is powered, it must be initialized manually due to safety reasons (which is common practice in robot automation industry). After initialization, the KUKA is switched to automatic run mode (PLC in control), where it acknowledges to the PLC that it is ready to operate. At this stage, the assembly devices as a whole (KUKA with the PLC) is ready to function. Here the KUKA remains idle until the HCoMS controller provides the PLC with a program number to execute. After a program number is issued, the KUKA uses the number to determine which subroutine case to execute and implements that behaviour. Figure 3.8 also shows an example of one of the subroutines that is executed when a product is quality tested. When a visual inspection is completed on a product, the HCoMS controller receives the test result from the vision station, and determines which operation it requires the KUKA to perform. Firstly, the HCoMS controller provides a program number for the KUKA to execute, then it provides the pass or fail information from the test results. Based on this information, the KUKA either remains idle if the product passed inspection or replaces the faulty product with another flawless product. On completion, the KUKA acknowledges to the HCoMS controller that the process is complete.

Figure 3.7 KUKA articulated robot arms

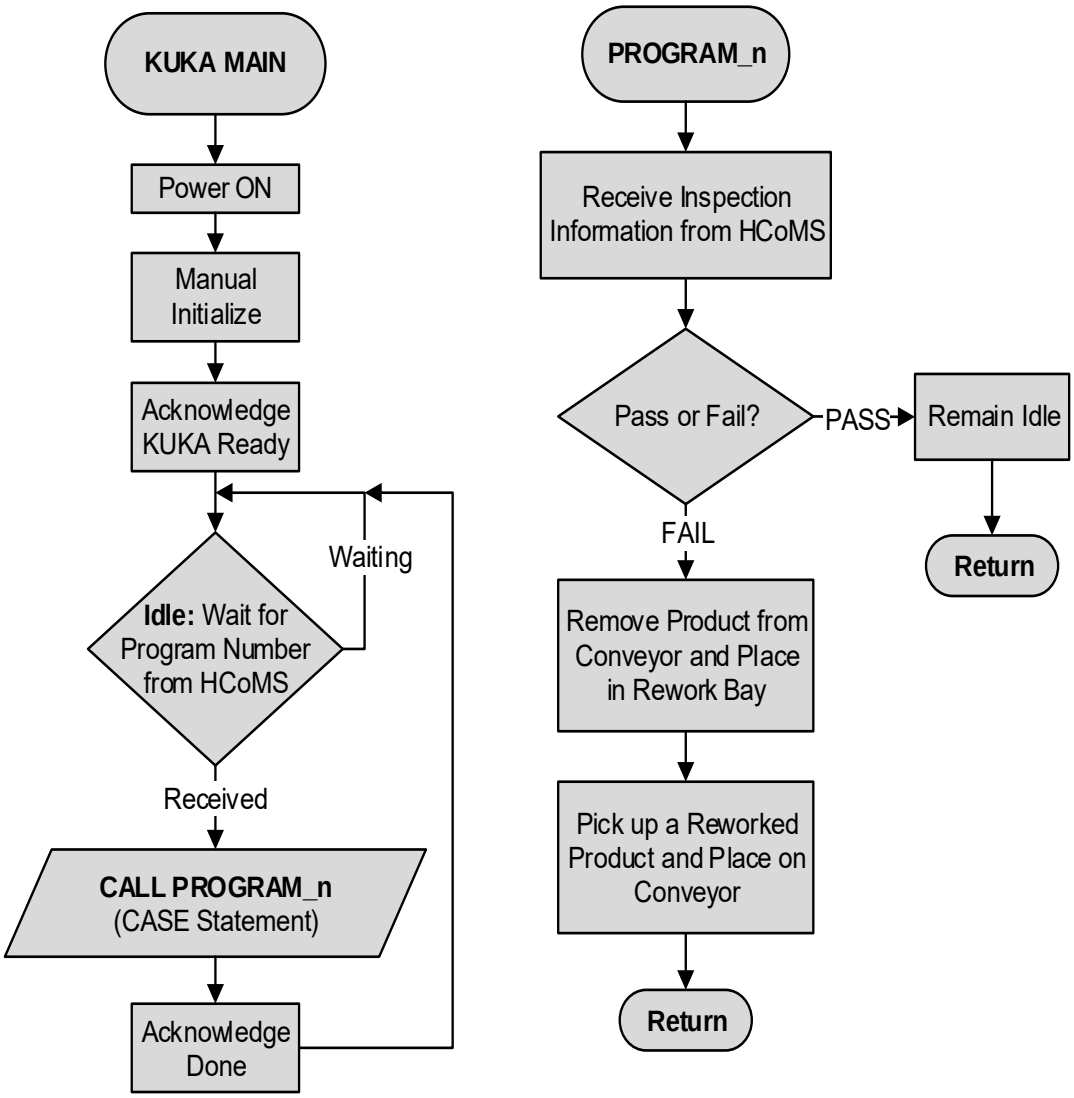


Figure 3.8 KUKA PLC Routine

3.3.4 Machine Vision Stations

The machine vision stations are used in the system to ensure the quality of the assembled products. These are modular stations with identical build, and are mounted at strategic locations in the system. This can be statically mounted at certain intersection points above conveyors, or mounted to a robot flange as an auxiliary tool (Figure 3.9) for flexible inspections in cases where either the robot will obstruct the camera view or there is a risk that the robot will be in collision with the camera. Figure 3.10 shows the generic routine of the machine vision stations. As the HCoMS controller handles a process for a product to be built, it will detect that the product is in place at an inspection point. The HCoMS controller will now provide parameters (information about what to test for) to the vision station and initiate a

start trigger. When the vision system receives the trigger, it visually inspects the product by capturing an image with a camera, and processing and comparing it to a reference image in memory that is selected based on the information provided by the HCoMS controller. After the visual inspection is complete, the vision system returns acknowledgement to the HCoMS about whether the inspection was passed or failed. If the inspection is passed, the HCoMS controller allows the product to proceed to the next process on the conveyor line. On the contrary, if the inspection was failed, the HCoMS controller signals a nearby KUKA to remove the product from the conveyor and place it on a conveyor that feeds the rework bay. Afterwards, the KUKA replaces the removed product with a repaired product from the rework bay (repaired product that passes inspection).

Figure 3.9 Cameras mounted as auxiliary tools

3.3.5 Mobile Devices

The HCoMS provides users with the option to interface with the system through the use of tablets and smart devices. This is achieved by using a software application developed for mobile devices, namely National Instruments (NI) Data Dashboard

for LabVIEW [61, 62]. NI Data Dashboard is an application that enables mobile devices to communicate with other computer-based applications that are developed using LabVIEW. It provides a user with the possibility to monitor network published variables, and view trends, historical data and alarms in a system. In addition, it facilitates returning control signals back to the LabVIEW developed applications. This, however, is not recommended for security reasons.

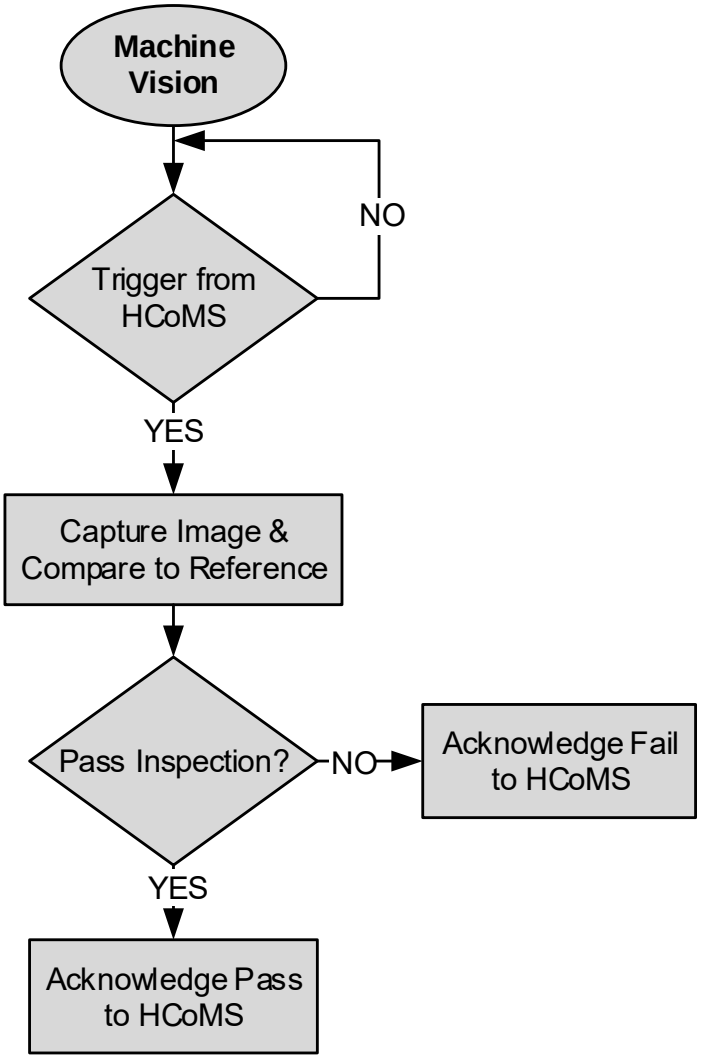


Figure 3.10 Machine vision stations routine

Data Dashboard allows users to build customised dashboard applications to monitor and control remote systems. First of all, a computer-based system must be developed using LabVIEW, where the variables used in this system are network published shared variables (shared variables represent type of global variable). Secondly, an application to monitor the system must be developed on a mobile device using Data Dashboard. At this stage, the Data Dashboard application

connects to the published variables and provides a portable view of the system developed in LabVIEW. An example of a mobile Data Dashboard application can be seen in Figure 3.11.

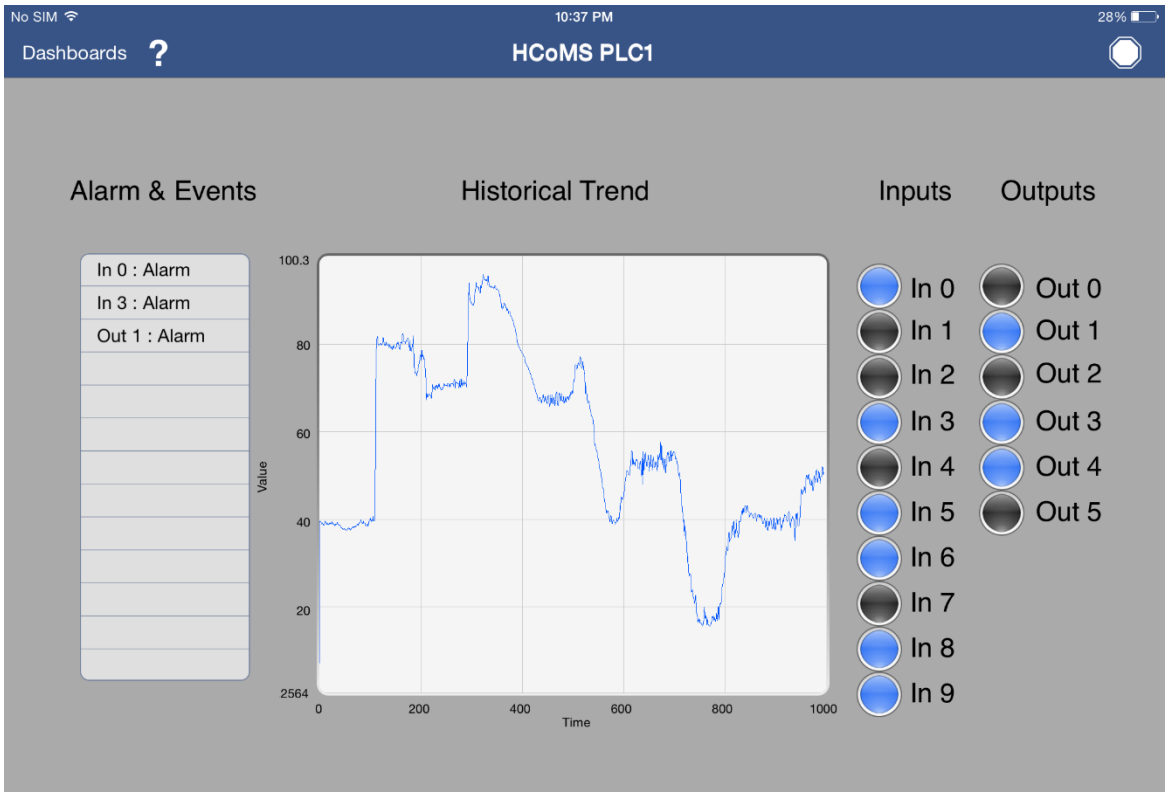


Figure 3.11 Data Dashboard example screen

3.4 HCoMS Software Architecture and Operation

This section provides an overview of how all the system software components are integrated as well as the preliminary once-off software. In addition, the ways in which all the software modules used in the system function separately and together are also explained.

3.4.1 System Design and Implementation Overview

An HCoMS is comprised of multiple software modules and functions along with various hardware components. The hardware components used in the system were discussed in detail in section 3.3. The architectural design of the system software however, can be better explained by referring to Figure 3.12. Firstly, Figure 3.12 shows a depiction of the main HCoMS Machine (physical PC) containing all the software, which is networked with all the external hardware components. Furthermore, it can be seen that the HCoMS main controller (application), the

shared variable engine (SVE) and the OPC server are the main software components that are integrated together within the HCoMS Machine. Next, focusing only on the HCoMS main controller, it can be seen how multiple software modules (which will each be explained subsequently), are integrated. The device installers are incorporated with the HCoMS main controller to perform various once-off auxiliary functions to aid in the operation of the system as a whole. These will be explained below.

It can be seen that external sales and purchase (can be from a remote PC) information are provided to the HCoMS controller to update the inventory (to keep stock of parts). Moreover, the main functional software modules in the HCoMS controller are the production planner, the production handler, the system configurator and the information manager. The production planner utilizes information from the detection and identification module (DIM); product recipe and device capability databases; as well as the inventory and ordering system to plan and schedule production runs. Furthermore, the function of the system configurator is to gather information from a user and perform automated configuring on the system based on this information. Additionally, the function of the production handler is to render the runtime front panel interface. It instructs the connected hardware devices to adapt its functional behaviour, as well as handle the real-time monitoring and control of each production run.

On the contrary, the information manager directly utilizes the SVE to perform its function. The information manager concurrently uses the Citadel database to log and retrieve production data from production runs, as well as incorporates the alarms server (DSC module) to send warnings and alarms about variables that are outside operating ranges. Additionally, the information manager also acts as the interface to communicate this alarm and log information between the system and the user. In addition, Figure 3.12 shows how the shared variable engine (SVE) is integrated with the HCoMS main controller and the OPC server. The function of the shared variable engine is to host the network published OPC-bound variables used by the HCoMS main controller and communicate these variables to the OPC server via an internal OPC client. This is achieved by creating and deploying runtime processes (also known as libraries) to map and host the shared variables. These

processes are essential for the information manager to operate. Collectively, all the software modules and functions integrated on the HCoMS machine allow for effective communication; monitoring and control between system components; and in entirety constitutes the HCoMS as a whole.

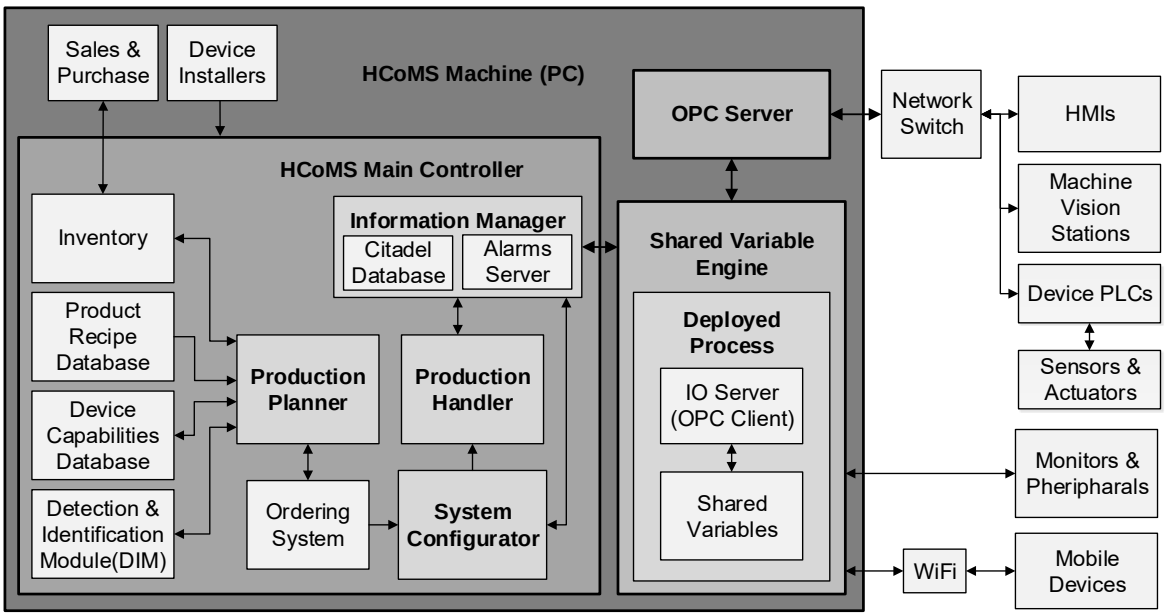


Figure 3.12 Overview of HCoMS software architecture

3.4.2 HCoMS Device Installers

An HCoMS has the ability to monitor and control any device that complies with a certain set of requirements. Referring to sections 3.2 and 3.3, all devices that can be controlled and monitored by an HCoMS must be modular and stand-alone in design; be able to connect to the system Ethernet network via an OPC compliant PLC; and have an HCoMS device installer developed for each distinctive device. These device installers are essentially software scripting programs that are similar to “patch” or “updating” software installers, and must be run once before the corresponding device can be used. After successful installation, an HCoMS will be able to recognise the relevant device (like an USB driver), have information about its functional capabilities (operations) and how to use it, as well as device setup (IO and server setup) and support files. An example of these files can be seen in Figure 3.13. When the device installer is run, the following process occurs: Firstly, the installer copies the function block used at runtime (device VI) and the “.csv” file for the OPC server setup to the correct file paths (locations). Next, the installer opens the system configuration file (.ini file), adds the device to the list of system

devices, updates the IO attributes, device address (IP address) and lists all the abilities and functions that the device is capable of performing. In addition, the installer also opens the system help file and appends the help information for the device to the file. Lastly, the installer opens certain sections of the system source code (VIs) and modifies it to accommodate the newly added device during runtime (create software references to the respective device).

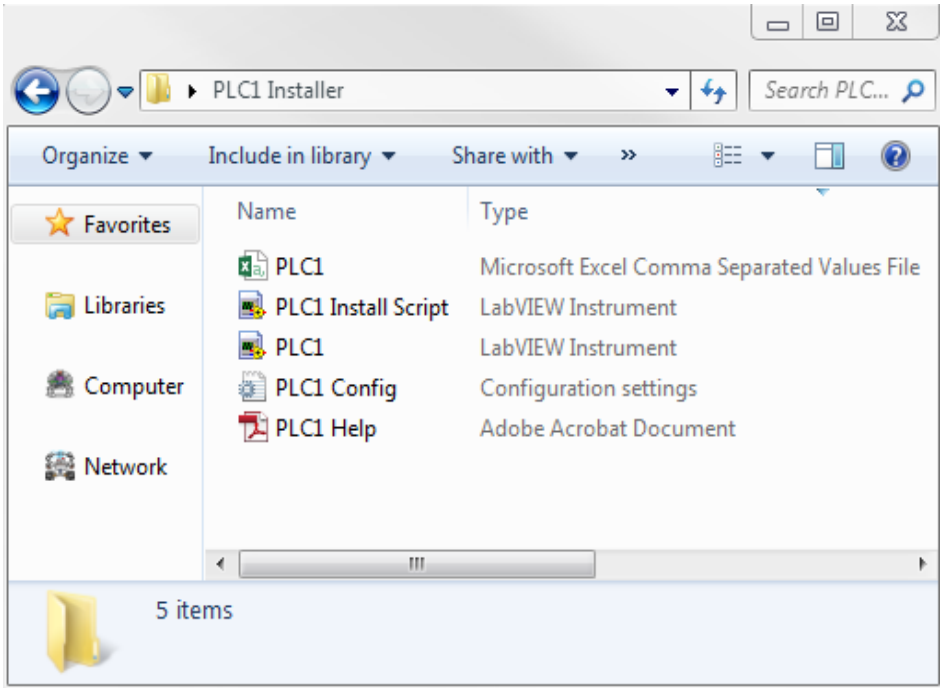


Figure 3.13 Files included with device installers

3.4.3 Software Modules

This section handles the software modules used in the HCoMS controller and how these operate as standalone software components. In addition, some of these modules are dynamic in nature, and will adapt their behaviour based on the chosen mode of configuration, which will be discussed subsequently.

3.4.3.1 Detection and Identification Module (DIM)

As the name implies, the functions of the detection and identification module (DIM) are to detect a network connection with a device and to identify the specific device that is connected. The DIM is a software module that runs continuously inside the HCoMS controller that detects which devices are connected to the HCoMS Ethernet network, as well as which devices have lost network connection. The devices which are connected and available on the network are listed and displayed in the DIM

interface. This can be seen in Figure 3.14 (Displayed in manual configure. Hidden in configure by product). In order for the DIM to function, the HCoMS device installers must first be run at least once for each of the devices that are to be used. This will ensure that the DIM will have the necessary information about the network address (IP address) and the device name (identity) of each device. If this step is completed, the DIM will successfully detect and identify each installed device connected to the network.

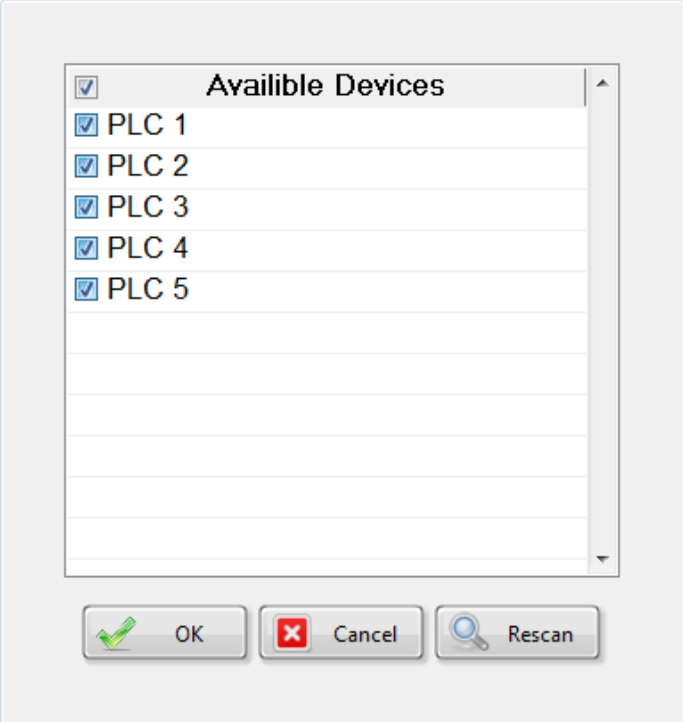


Figure 3.14 Detection and identification module showing connected devices

After all the desired devices are added and installed to the system, the system is ready to be utilized and operates as explained next. Initially, the DIM “pings” each device to check if a connection can be established between the device and the HCoMS controller. This is initially the quickest method to obtain all the devices connected to the network, whereafter these devices are relayed to the production planner to determine a production plan and schedule. This information is used to initially configure the system.

On the other hand, the DIM redundantly uses two concurrent methods to keep checking network communications during runtime. The first method that the DIM incorporates is what is known in industry as a “heartbeat” connection. This is a simple but effective method for both the HCoMS controller and devices to detect

that the connection is uninterrupted and enables each to react to the interruption. Fundamentally, the DIM sends a signal to each device by altering a variable (e.g. setting a bit). Afterwards, if the respective device realizes this change, it will acknowledge communication by altering the same variable. Both entities, will keep monitoring for a change within a certain period of time. If a timeout occurs (like a watchdog timer), both entities will realize a connection lost with the other. An example of the code can be seen in Figure 3.15.

For the second method, the DIM performs cyclical reads from a variable in each device (similar to the heartbeat connection) to inspect the signal quality of the variable in the OPC server. This is achieved by opening a connection, waiting for the connection to be made (delay), performing a read from an OPC variable in each device to check the signal quality, and then closing the connection. This is shown in Figure 3.16. From the viewpoint of the HCoMS controller, this will establish whether communication is lost for a single device or multiple devices. Thus, this method will provide a good indication of where the connection error occurred (device, HCoMS controller or OPC server). In addition, this method allows the DIM to communicate these failures to the information manager and inform the user by causing “bad signal quality” alarms if the signal quality is weak or bad. Although the DIM uses redundant concurrent methods to detect and identify devices connected to the network, it ensures that communication lost is successfully detected and mistakenly detected communication lost, is eliminated. Furthermore, this redundancy does not affect the performance of the system regarding processor overhead or increased latency on the communication network.

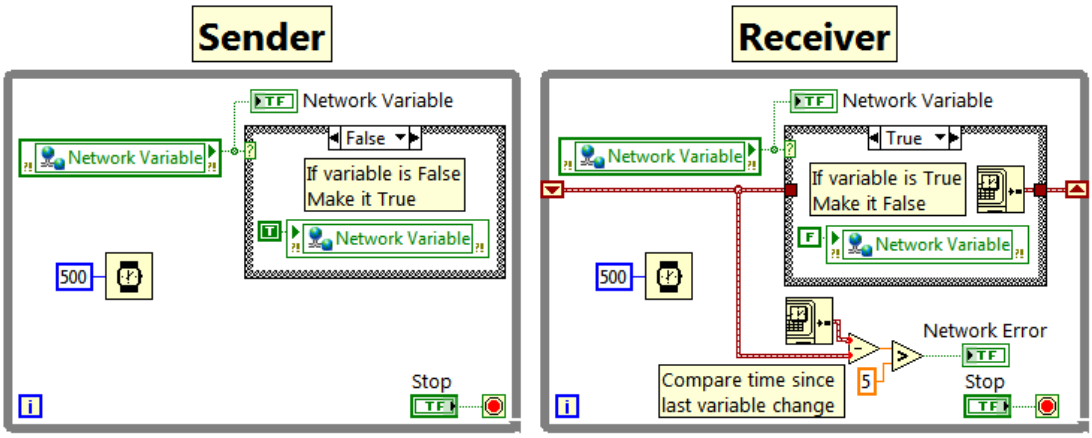


Figure 3.15 Example of heartbeat communication code

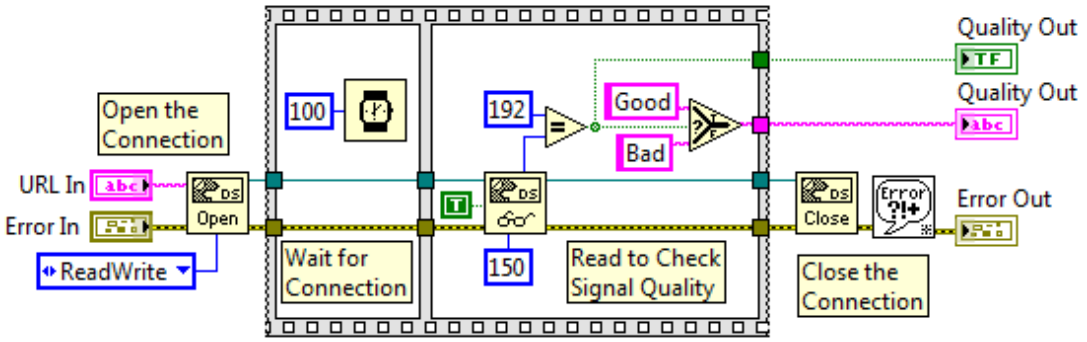


Figure 3.16 OPC signal quality check

3.4.3.2 Information Manager

The information manager is a software module that performs functions like handling runtime data (updating the front panel), system alarms and data logging (and retrieval). To achieve this, the system utilizes a shared variable engine (SVE), which is inherently installed with LabVIEW, to run deployed processes also known as libraries. These deployed processes are required to host the network published shared variables that are created for each system device, where these variables are communicated to the OPC server via an internal OPC client to control the system devices.

With this in mind, the information manager acts as a software client to enable users to access and manage system variable information from the SVE. Excluding the fact that the information manager updates runtime information on the front panel, it also acts as an interface for users to retrieve and view logged information as well as manage system alarms on the SVE. Essentially, the information manager encapsulates methods to access the alarms server (from the DSC module) and the Citadel database, to provide users with this interface to retrieve and manage the information on the SVE. The information manger updates variables shown on the front panel and allows users to launch detailed views of system devices, system alarms and historical data (trends), which will now be discussed.

Figure 3.17 shows an example of the client front panel during runtime as it updates the current state of IOs for all the devices. When device alarms occur during runtime, the grid positions containing the respective devices that are in alarm, will be highlighted (and flashing) red, as seen in the figure. In addition, a summary of all the devices in alarm are displayed in a banner just below the positions grid (Figure 3.17).

Notice that each alarm has an icon (pictogram) with a different shape and colour to indicate the priority of the alarm. Furthermore, the text colour in which each alarm in the list is displayed differs, depending on their acknowledged states.

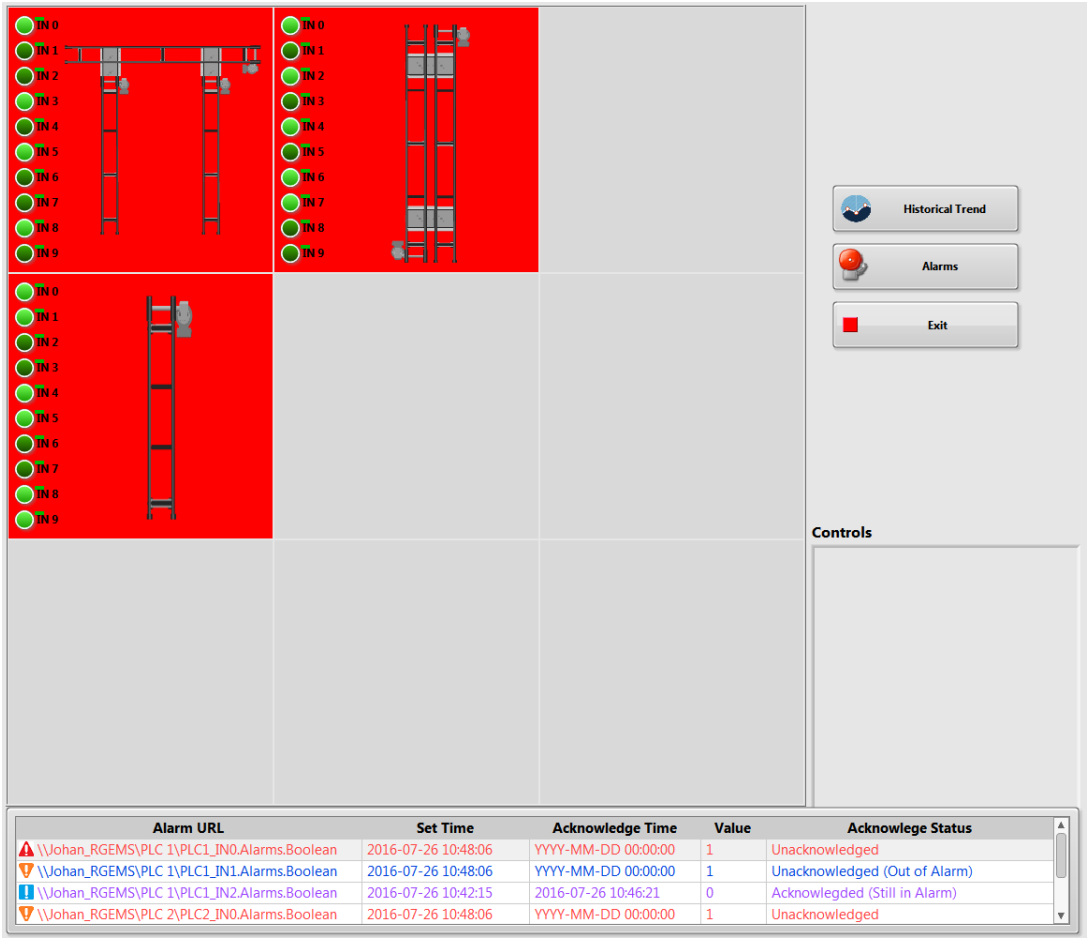


Figure 3.17 Alarms on front panel

To manage these alarms, a user can either launch a detailed view of the respective device in alarm or launch the alarms summary detailed view, which contains a list of all the alarms of all the devices in alarm. To launch a device detailed view, the user simply has to click on one of the respective devices shown on the front panel. The device detailed view opens in a separate window, which contains the current state of the IOs for the selected device and a summary of the alarm conditions. At this time, the user can acknowledge any alarm that occurred by selecting it in the list and clicking the “Acknowledge” button. Instead, the user can simultaneously acknowledge all the device alarms by clicking the “All” button. Afterwards, the user can close the device detailed view by either clicking the “Exit” button or simply exiting the window. An example of a device detailed view can be seen in Figure 3.18.

Alternative to launching a device detailed view, the user can also manage alarms by opening the alarms summary detailed view window. This window can be accessed by either clicking on the alarms summary at the bottom of the front panel (banner below the grid), or simply clicking on the “Alarms” button. The alarms summary detailed view contains a list of all the alarms for all the devices in the system and can be seen in Figure 3.19. At this point, the user can acknowledge any of the occurring alarms in the same manner that was used for the device detailed views.

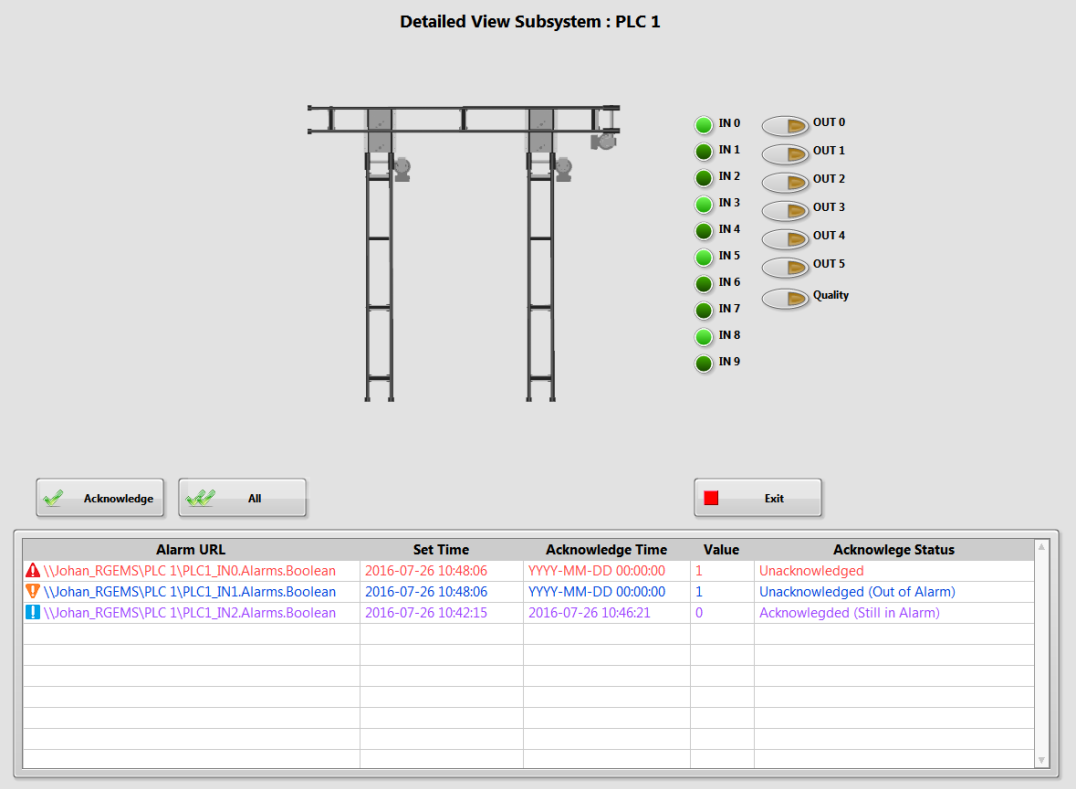


Figure 3.18 Device detailed view

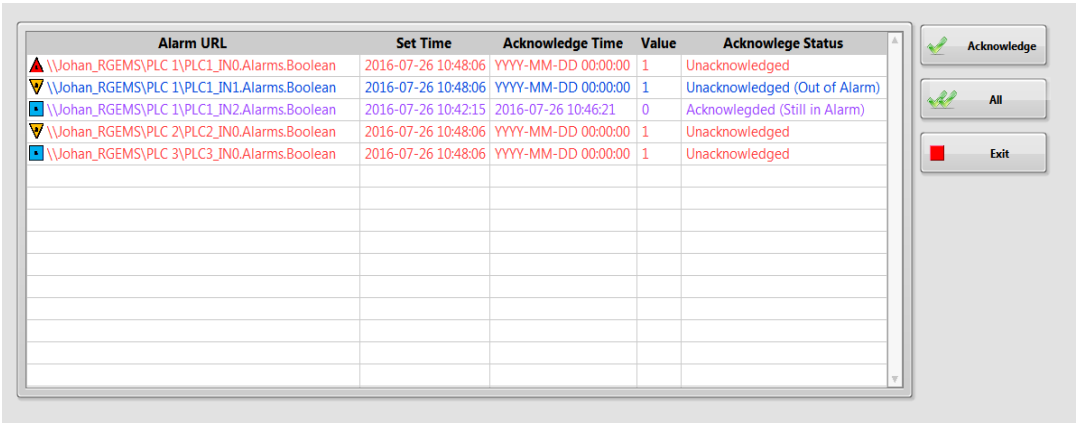


Figure 3.19 Alarms summary detailed view

In contrast to displaying runtime data and managing device alarms, users can retrieve and view historical data by launching a detailed view of the historical trends. The user can launch the historical trend detailed view by clicking on the “Historical Trend” button and then filtering the information by device and time logged (date and time). An example of the historical trend detailed view can be seen in Figure 3.20.

To retrieve and filter data from the Citadel database, users can start by selecting a desired device from the dropdown menu, which will contain all the devices added to and present in the system. The historical trend detailed view will update to display the information for the chosen device. In addition, users can also browse through the device information by defining the date and time that the specific process ran (was logged). This can be done by either scrolling through the time by using the scroll bar, selecting a start and end timestamp (by timestamp), or by using the graph tools to zoom to the required date and time selection of the desired information. In addition, the displayed information can further be modified by hiding unwanted device variables by selecting or deselecting it on the legend that shows all the available IOs for the selected device. After users acquired the desired information, either another device can be selected or the detailed view can be closed.

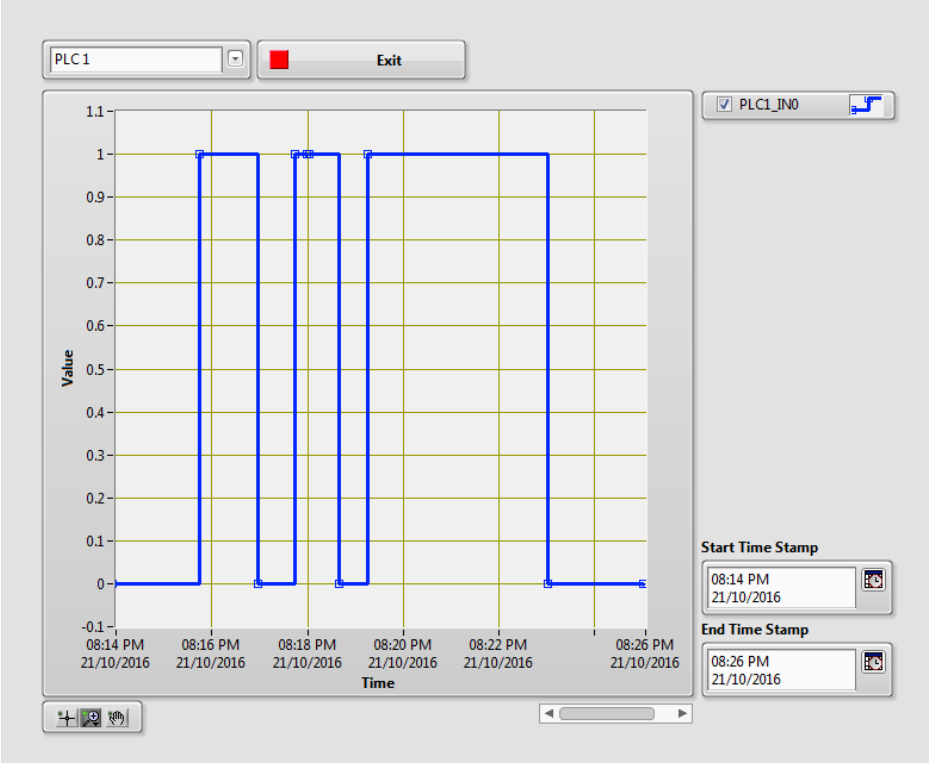


Figure 3.20 Historical trend detailed view

3.4.3.3 Ordering System

In attempting to build or assemble products, product orders must be placed using the ordering system. The ordering system is a standalone software module that is integrated with the production planner and system configurator. The behaviour of the ordering system depends on which configuration mode was selected from the main menu (Figure 3.31) and presents the user with a different front panel interface for each. This can be seen in Figure 3.21 and Figure 3.22 which shows the ordering screens for “manual configuration” and “configure by product” modes respectively. The reason for this difference is that the HCoMS automatically determines the system configuration in “configure by product” mode (user only needs to choose a product), where in contrast, the user can specify a system configuration to accommodate numerous products using the “manual configuration” mode. When the user is presented with an ordering screen (don’t matter which mode), the user can select from products to build along with the required quantities for each in the box at the top of the screen. Notice that the “configure by product” ordering screen does not display scheduling information or allow the user to rearrange orders. In contrast to the “configure by product” ordering screen, the user can repeatedly add orders to the production queue in “manual configuration” mode. The ordering system will now send the new order data to the production planner, where it will determine a production plan and schedule and return whether the updated order can be performed. In addition, the user can select an existing order in the order screen and choose to remove it from the queue. Furthermore, orders can be moved up or down the queue, by selecting it, and clicking the “Move Up” or “Move Down” buttons. Notice in Figure 3.21 that the additional information also provided to the operator include the duration of time the order will take to complete; the starting time and the estimated time of completion; the date of completion as well as the current status of the order. Worth mentioning, the ordering system utilizes globally network published shared variables which allows it to run either on the same PC as an HCoMS, or on an external PC if required.

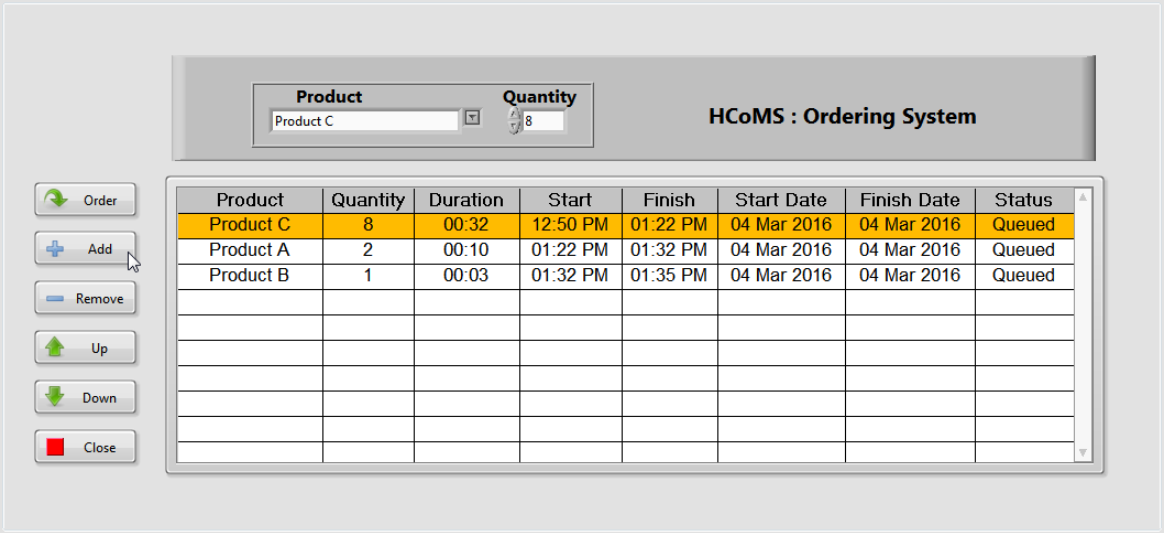


Figure 3.21 Ordering screen during manual mode

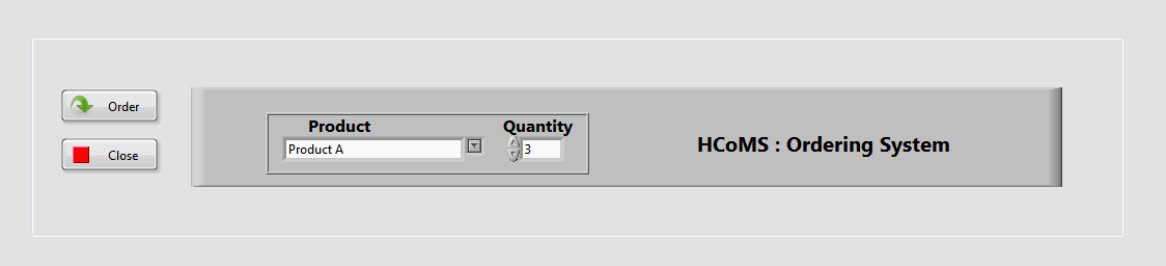


Figure 3.22 Ordering screen during configure by product mode

3.4.3.4 Production Planner

The production planner is a software module that performs system functions similar to the ERP and APS systems that were discussed in Chapter 2. Fundamentally, the production planner is a “brute force” solver that determines if an order that is placed can be executed, when this will happen, and with which resources it will be performed. It uses a set of rules and conditions, along with the information received from the software modules integrated with it, to solve a “best fit” production plan.

The functionality of the production planner can better be explained by referring to Figure 3.23. Once a user places an order in the ordering system, the ordering system will format the information entered by the user and send it to the production planner. The ordering information will contain which products and how many of these to assemble. Next, the production planner acquires information from the inventory, product recipe and device capabilities databases. Afterwards, it obtains which devices are connected to the system and uses the previously attained

information to start the planning procedure. Firstly, it inspects each product recipe to realise which parts are needed to assemble the product and which processes must be followed to do so. It then compares the required parts to assemble the products with the amount of parts available in the inventory. If the available parts are enough, the production planner will compare the processes to assemble a product with the capabilities of each device that are connected to the system. At this point, the production planner will "brute force" solve whether a production run can be made. If possible, the production planner will schedule the sequence in which the products will be assembled based on the time requirements for each from the product recipe and send this information back to the ordering system.

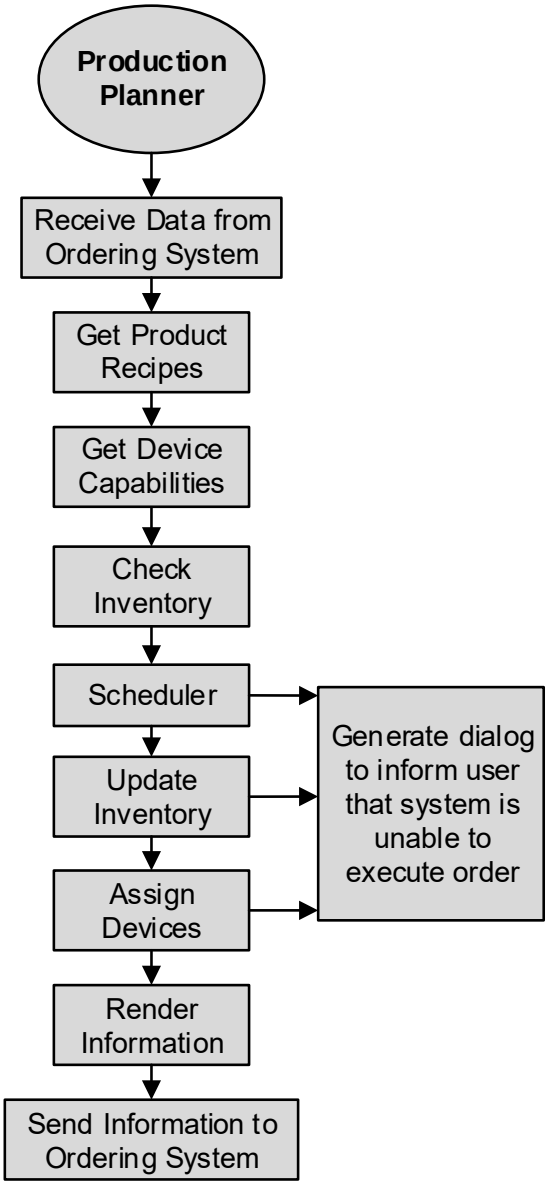


Figure 3.23 Functionality of the production planner

In contrast, if a production plan cannot be solved, the production planner will send back this information to inform the ordering system that it is not currently possible to perform the production run. At this point, the user has an opportunity to adjust product orders or resolve the factors that prohibits production (e.g. add parts to the inventory or connect the required devices). Afterwards, the production planner will again determine whether a production run is possible and send this information to the ordering system.

3.4.3.5 System Configurator

3.4.3.5.1 Overview

The system configurator is a software module that performs multiple subtasks and functions based on which system configuration mode is chosen. Firstly, it gathers configuration information from the user and utilizes it to manage and oversee the system configuration process. The system configurator accomplishes this by launching the automated configuration routine, which will be discussed in the next section. Furthermore, it is also responsible for generating the sections of source code that specify how system hardware are interconnected and used during runtime in production runs. The behaviour of the system configurator largely depends on which configuration mode was selected from the main menu when the application was started. The main executions in each mode selection are mainly the same, but have minor deviations that will be discussed in detail in subsequent sections.

The first task that the system configurator performs after product orders have been placed and the production planner determined a production run, is to configure the system. At this point, the system configurator will prompt the user about desired configurations and if modification should be made to the recommended IO attributes of the system (for alarm and logging etc.). The system configurator will first prompt the user to decide which of the detected connected devices to include in the system configuration. Next, the user is prompted to choose how and where these devices are displayed on the runtime front panel by specifying grid positions and orientations for each. In addition, the user can choose to alter the IO attributes for each device connected to the system. The IO attributes that can be configured for each device include whether alarming conditions are enabled or not, as well as specifying the

data logging requirements for each device. In addition, the set points (values) for alarm conditions can be specified, and also if these conditions should be logged.

After all the required configuration information from the user is captured, the system configurator starts the system configuration process by performing the automated configuration routine, which will be explained thoroughly below.

3.4.3.5.2 Automated Configuration Routine

The automated configuration routine utilizes the information acquired from the user to perform the system configuration automatically (automated). The routine performs a series of tasks (subroutines) to set up preliminary elements that are required to run the system, and also to create and set up the variables for each device connected to the system. The first task that the automated configuration routine performs is to set up the Citadel database. The routine searches for an existing database and appends newly added information or creates a new database if it does not exist.

After the database is set up, the automated configuration routine creates the processes on the SVE to host the shared variables of each device. After the processes are created, the network published shared variables for each device are created, and then deployed to the SVE (Figure 3.24). Next, the alarm and data logging information for each device are updated in the Citadel database.

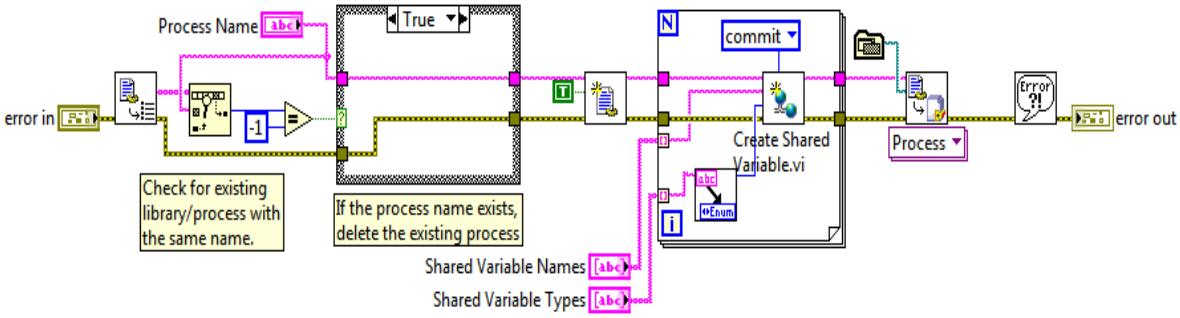


Figure 3.24 Creating processes and shared variables

At this stage, all the shared variables for each device are created, hosted and configured. The next task that the automated configuration routine executes is mapping (binding) the shared variables of each device to the specific physical device IOs (PLC IOs) on the OPC server (Figure 3.25). This ensures that the

physical devices can be controlled directly from the real-time LabVIEW application. For the last task, the automated configuration routine opens the front panels of each device and maps (binds) the same shared variables (device shared variables) to the respective front panel control and indicator objects (Figure 3.26). This results in, if any device IO changes, that the corresponding front panel object will change accordingly. This is the key that enables the information manager to relay information to the user.

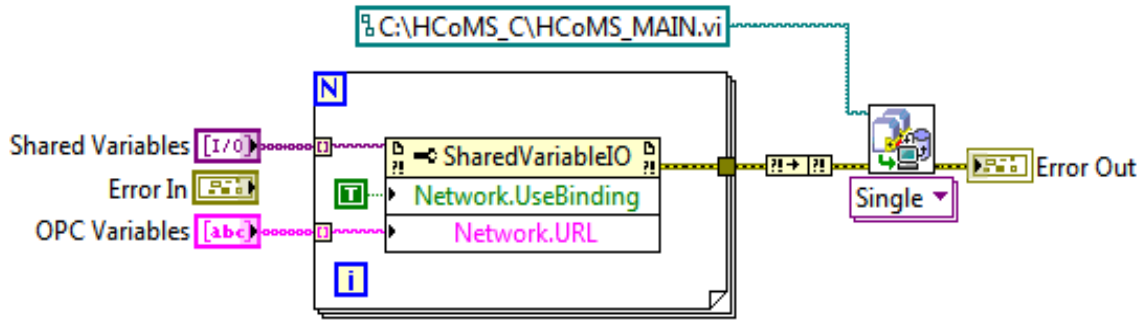


Figure 3.25 Mapping shared variables to OPC server

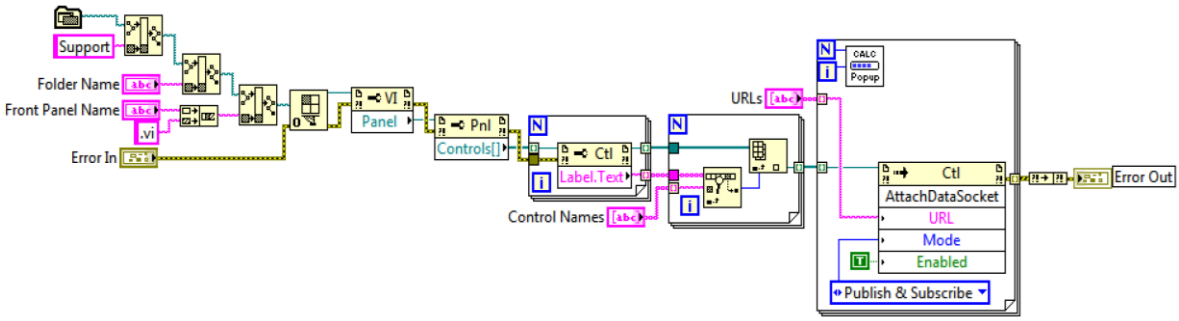


Figure 3.26 Mapping shared variables to front panel objects

After all this, the automated configuration routine is complete and the bulk of the system is configured. At this point, the system configurator must generate the code in the background that defines how the system devices are interconnected. This will be discussed in the subsequent section.

3.4.3.5.3 Code Scripting and Virtual Wiring

As stated in the previous subsection, the next task that the system configurator must complete is to generate the background code used during runtime that specifies how the system devices are interrelated. The system configurator generates this runtime code by making use of VI scripting [63-65] and applying the concept of virtual wiring.

The virtual wiring interconnects the system devices together in the software. Therefore, the user does not need to physically wire to the system devices together. To clarify, the system utilizes a block diagram (VI back panel) in the LabVIEW IDE as the coding environment to develop (write) the runtime code. This is equivalent to how a software developer would write LabVIEW code (firmware), but the code development is managed by the system itself instead of the developer. With this in mind, the system virtually connects the IOs of devices by wiring the respective VIs of each device together in the block diagram. Since these IOs are also mapped to OPC, this makes a connection similar to physically wiring devices together.

How the runtime code is scripted depends on which configuration mode the user has selected. If the user selected the “configure by product” mode, the system configurator will open a block diagram template, populate the block diagram with the VIs of each device in the configuration, and automatically wire the VI connections together. After this, the system configurator will save the block diagram as runtime code in the required file location and close it. On the contrary, if the user has chosen the “manual configuration” mode, the system configurator will open a block diagram template, only populate the block diagram with the VIs of the configured devices and present the user with the window shown in Figure 3.27. At this stage, the system allows the user to open the block diagram of the presented window, arrange or rearrange the VIs, and wire together the VI connections as desired. On completion, the user simply has to exit the window, and afterwards, the system configurator will save the block diagram as runtime code in the vital file location. An example of the runtime code can be seen in Figure 3.28. At this point in time, the system is fully configured and ready to perform a production run, which will be discussed in the subsequent section.

If it is found that the runtime process (production run) operates erroneously due to the possibility that the system devices are virtually wired wrong, the runtime process can be stopped and the connect devices window (same window as in in Figure 3.27) can be launched. At this time, the mistakes can be rectified by correcting the wiring between the device VIs. On completion, the connect devices window can be saved and closed; and the runtime process can be run again. This procedure can be repeated as many times as required.

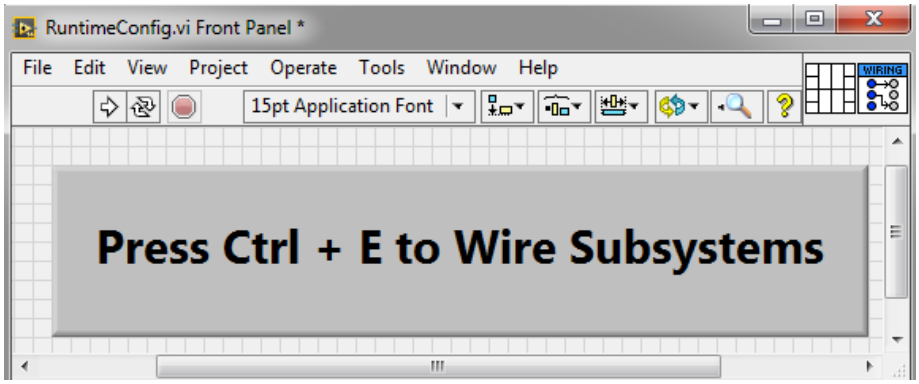


Figure 3.27 Dynamic virtual wiring front panel

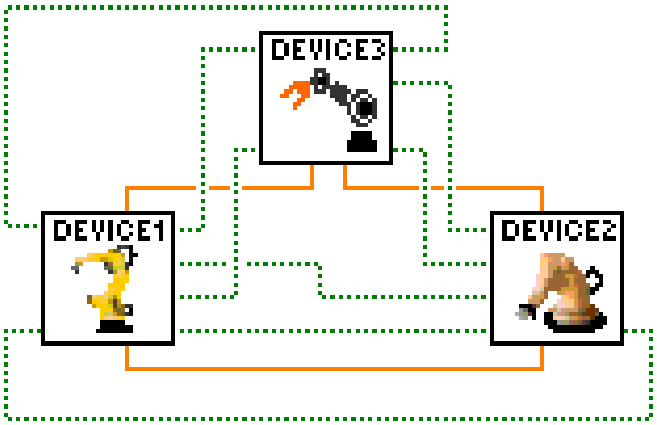


Figure 3.28 Example of block diagram code

3.4.3.6 Production Handler

The functions of the production handler in the system is to render the front panel display during runtime and to manage the monitoring and control of production runs. When the system configuration is done, the user can open the front panel of the runtime client and perform a production run.

Upon opening the runtime client, the production handler utilizes the information that the system configurator acquired from the user to render the front panel. Depending on the mode selected, the production handler displays each configured device in a grid position as well as orientation that was specified by the user (manual configuration). An example of this can be seen in Figure 3.29. On the contrary, if the configure by product mode was chosen, the production handler will automatically decide what the grid positions and orientations should be and render the front panel as such. At this stage, the front panel shows an overview of the entire configured system (top view) and where each device is placed within the system. In addition,

the front panel shows how each respective device is functioning in real-time (current state of device IOs) and allows the user to open a detailed view of it by clicking on it on the front panel. To clarify, the information that is displayed on the front panel and in the detailed views are obtained from the information manager discussed previously.

At the same time that the front panel is initially rendered, the production handler informs the hardware devices connected to the network to adapt its functional behaviour. Since the devices can have multiple behaviours, it adapts its behaviour based on program numbers that are allocated to it from the production handler. In addition to device behaviour, the production handler also manages the production run background processes. This includes supervising the assembly process, and keeping track of product quantities that have been made, as well as how many must still be made.

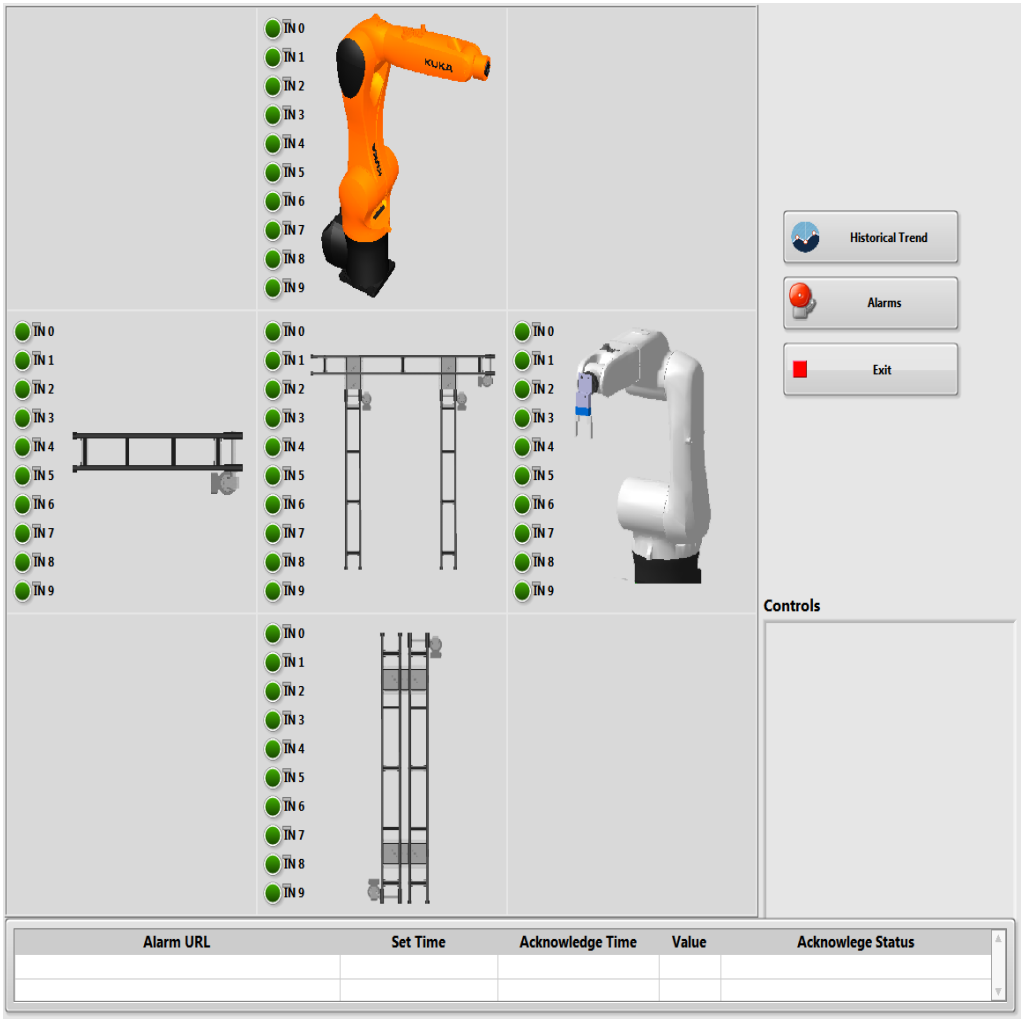


Figure 3.29 Front panel rendered

3.4.4 Software Functional Operation and Configuration Modes

3.4.4.1 Software Functional Overview

The HCoMS main application is developed to assist a user by configuring the system automatically (automated), and monitor and control the system during runtime. The software modules discussed in the previous sections collectively form an HCoMS as a whole, and how the system (HCoMS) functions are discussed next. The overview of the system functional operation can be better explained by referring to Figure 3.30. Firstly, notice that Figure 3.30 includes differently coloured legends that depict the different functional paths that the software can execute. In addition, notice that some of the different paths use the same functional blocks. These functional blocks are dynamic routines that adapt its specific behaviour based on the path chosen by the user (changes behaviour bases on mode it is used in).

At start up, the HCoMS main application presents the user with the main menu screen shown in Figure 3.31 and waits for initial instructions from the user. At this time, the user can choose from the various methods (modes) to configure the system. These modes include “manual configure”, “configure by product”, “load configuration” and “save configuration”. When a method is chosen, the software will execute that specific configuration mode by capturing the required information from the user, performing the required configurations, prompting the user with instructions (if any), and then running the production process.

After production runs are finished, the system can either run another production batch with the same configuration, or reconfigure the system with another desired configuration mode. If the system does not require reconfiguring, the system will check the product order information, if the system configuration is still valid (are devices still connected?), and then run the production process. Otherwise, the system requires to be reconfigured by utilizing one of the configuration methods (modes). Each configuration method (mode) functions differently and are used with certain aims in mind. How each mode available in the system functions will be explained in detail in the following sections.

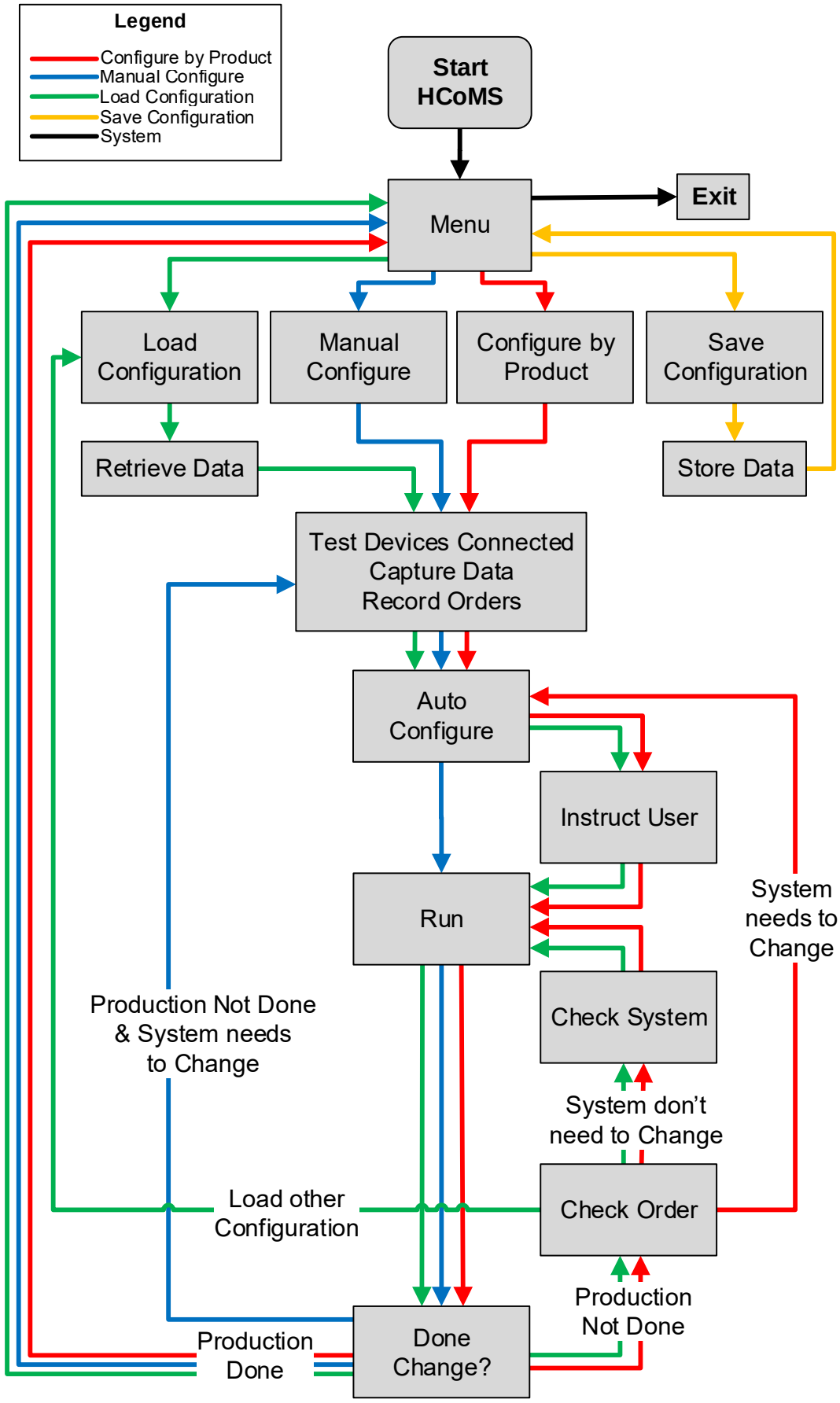


Figure 3.30 HCoMS functional operation overview

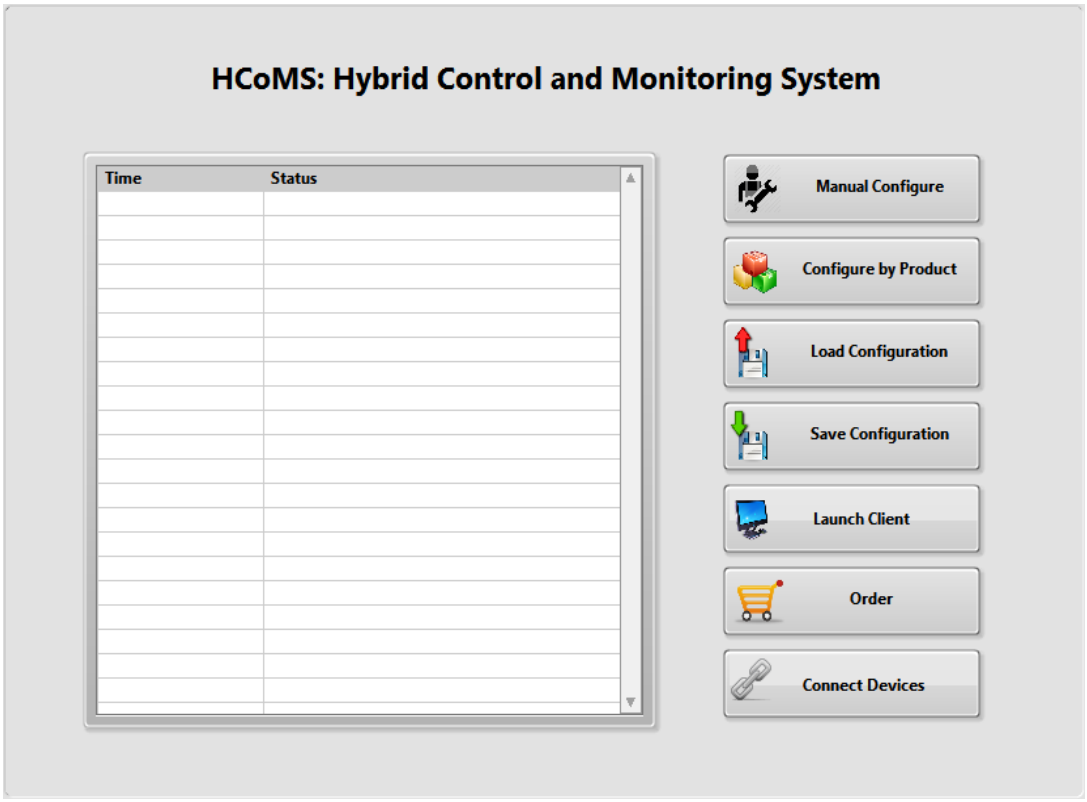


Figure 3.31 Main menu screen

3.4.4.2 Manual Configuration

The intention of the “manual configuration” mode is to allow a user full control regarding the system configuration. This mode is used when a user has knowledge of what the desired physical configuration should be. In addition, this mode assists the user in the system configuration by performing automated configuring routines, but still allows the user to make the final decisions about how to configure the system.

The detailed operation of the “manual configuration” mode can be better explained by referring to Figure 3.32. During system runtime (when the HCoMS application is started) the user can initiate this configuration mode by clicking the “Manual Configure” button from the main menu (Figure 3.31). At this stage, the HCoMS will scan which devices are currently connected to the system network (detect devices); present the user with a list of these available devices (Figure 3.14); and allow the user to select which of these devices to include in the system configuration (some or all of them). After the devices are selected, the user is presented with a window shown in Figure 3.33 to select where on the runtime front panel the corresponding device must appear and in which orientation the device must appear (grid position

and orientation). Furthermore, the user can click the “Help” button to launch a PDF help file to assist in deciding these orientations and grid positions. This will represent the physical layout of the system devices during production runs. At this stage, the HCoMS concurrently launches the ordering system and the production planner. The user can now utilize the ordering system screen to place product orders.

While the product orders are placed, the ordering system relays this ordering information to the production planner. The production planner now determines whether it is possible to perform the order with the available resources, schedule it as such and reply to the ordering system. After the user is satisfied with the product orders, the production batch order can be placed by clicking the “Order” button in the ordering screen (Figure 3.21). At this time, the HCoMS provides the user with an opportunity to modify the IO attributes of each device. After this, the HCoMS has captured all the data required for configuring the system and starts the automated configuration routine.

The sequence in which functions are performed during the automated configuration, is as follows: creating or updating the Citadel database; creating processes (libraries) for each connected device; creating shared variables for each device; mapping these shared variables to the OPC server; mapping these shared variables to device front panels; and deploying the device processes to the SVE. After the automated configuration is completed, the HCoMS generates the runtime code for the user to wire together the system devices. When the user is done, the runtime code (VI block diagram) can be saved, closed and then be used during runtime. At this point, the system is fully configured – and the user can then launch the runtime front panel to monitor and control the production run. After a production run is complete, the HCoMS will check if the entire production batch are finished. If not, the HCoMS will determine whether the current configuration is still valid (devices still connected) and perform the next production run.

On the contrary, if the configuration must change, the HCoMS will prompt the user again to acquire the new configuration information and repeat the automated configuring process. If the entire production batch is completed, then the user can decide to either exit the HCoMS application or return to the main menu to perform other production runs with other system configurations.

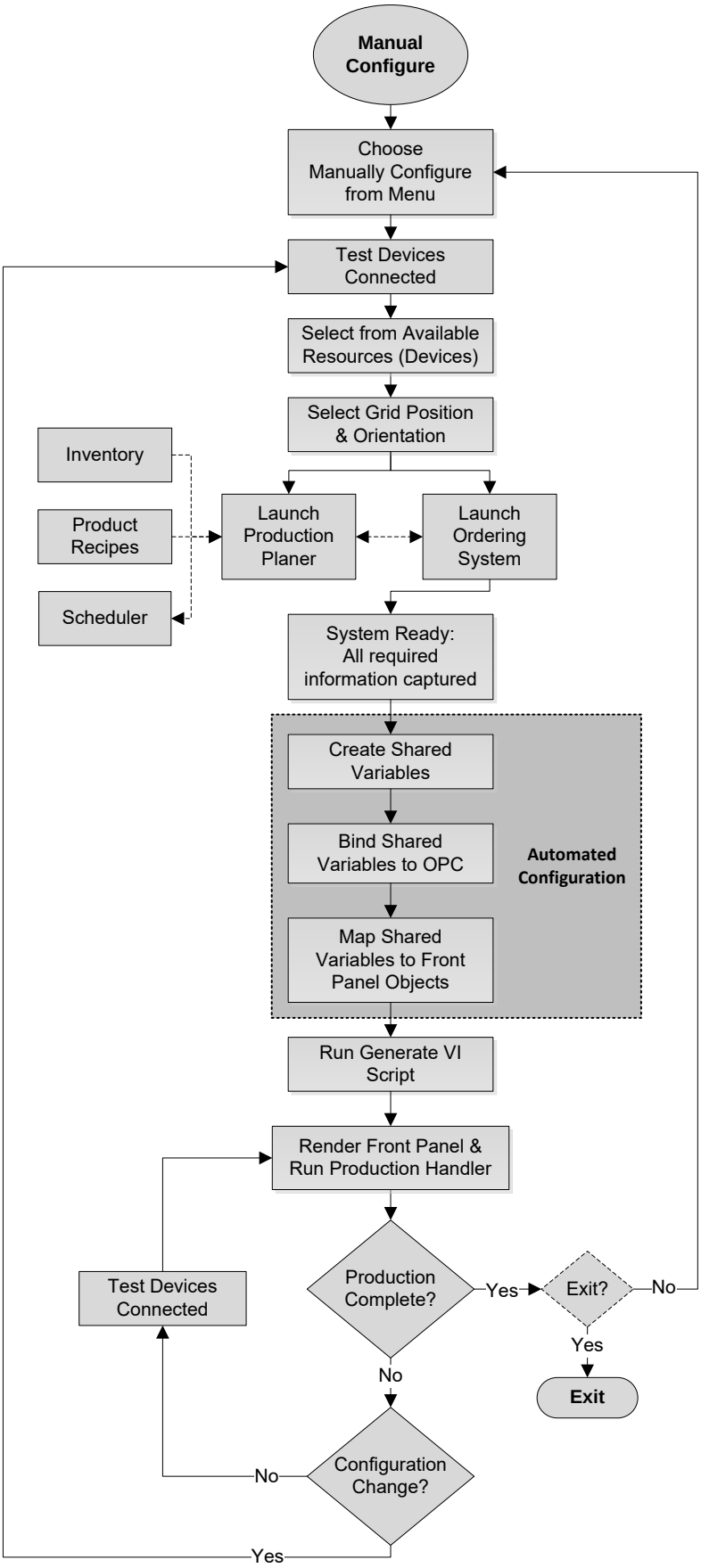


Figure 3.32 Detailed operation of the manual configuration mode

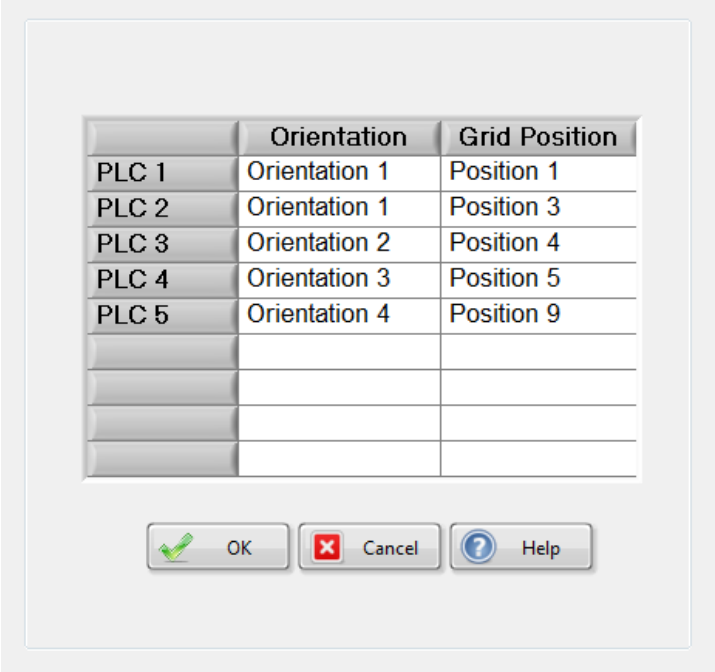


Figure 3.33 Grid position and orientation selection

3.4.4.3 Configure by Product

The objective behind the “configure by product” mode is to automate the configuring process as much as possible, with minimal input from the user. This mode is used when only one product type is assembled; the user is uncertain about how to configure the system; or the user do not want to restrain the system to a precise physical layout. The system will configure with the system-recommended attributes and instruct the user about the physical layout. This method requires less information from a user and performs the configuration process quicker, but the user has less control over the actual configuration and physical layout. The detailed operation of the “configure by product” mode can be explained by referring to Figure 3.34.

The user can initiate this configuration mode by selecting it from the main menu screen. Similarly to the “manual configuration” mode, the HCoMS launches the ordering system and the production planner. Here the user has to specify only which product to assemble and the quantity thereof. The production planner will then determine if it is possible to perform the current order with the available resources and then initiate the configuration process if it is. At this point, the HCoMS will determine the best configuration for the system and select the front panel positions and device orientations automatically.

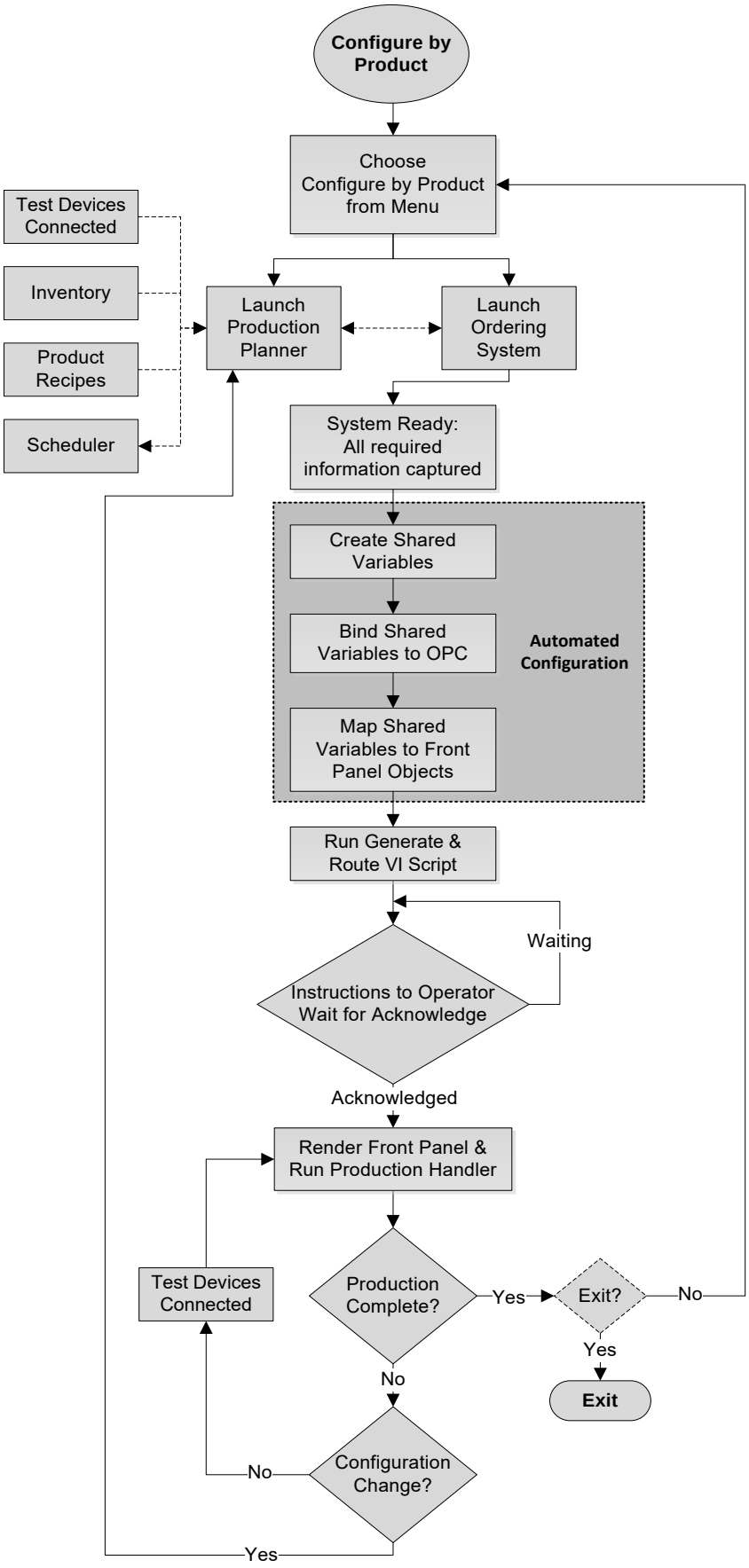


Figure 3.34 Detailed operation of the configure by product mode

After this, the HCoMS uses the acquired information and performs the automated configuration in the same way as it did for the “manual configuration” mode. After the automated configuration is completed, the HCoMS generates the runtime code, but in contrast to the “manual configuration” mode, it automatically routes the system devices together, and then saves and closes the runtime code (VI block diagram). The HCoMS then conveys instructions to the user about the physical layout of the system and waits for acknowledgement that it is complete. From this point onward, the system will operate exactly the same as in the “manual configuration” mode. The user can launch the runtime front panel and perform the production run; and after the production run, decide whether to reconfigure the system, return to the main menu to choose another configuration mode or exit the HCoMS application.

3.4.4.4 Save Configuration

At any stage after a system configuration or production run, the user can decide to save the current system configuration (if it is a commonly used configuration) so that it might be used again later (Figure 3.35).

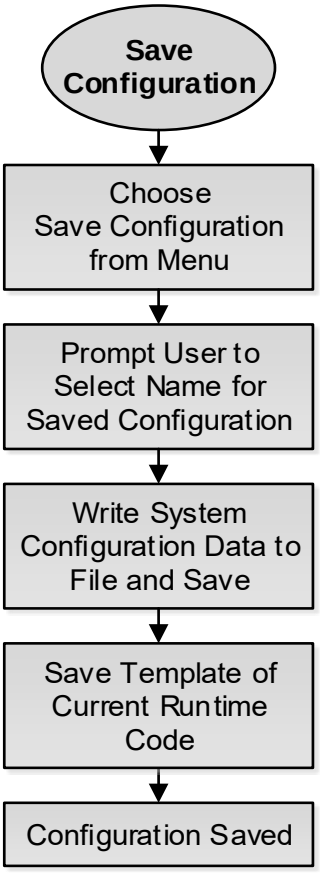


Figure 3.35 Detailed operation of the save configuration mode

The user can initiate the “save configuration” mode by selecting it from the main menu. Afterwards, the user will be prompted to provide a name for the saved configuration. The HCoMS will then write the current system configuration and IO attributes to a configuration file and save it. In addition, the HCoMS will also save a copy of the current runtime code (VI block diagram). When it is done, the HCoMS will return to the main menu and wait for further instructions from the user.

3.4.4.5 Load Configuration

The “load” configuration essentially complements the “save” configuration. The intention of this mode is to load a previously saved configuration and determine if the configuration can be achieved with the connected devices (Figure 3.36).

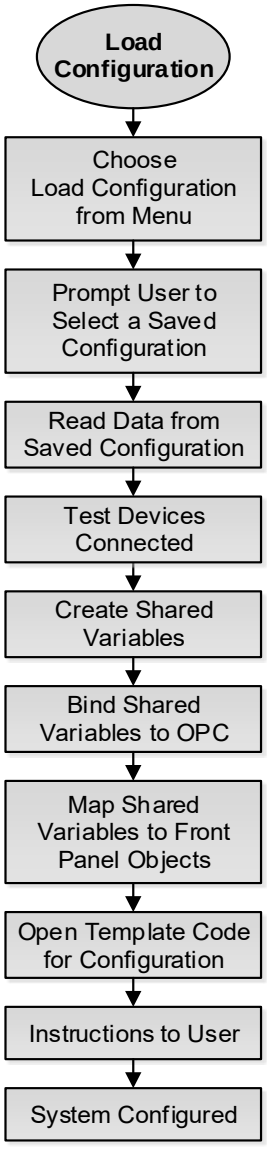


Figure 3.36 Detailed operation of the load configuration mode

Referring to Figure 3.36, the user can load a configuration by selecting this mode from the main menu. The HCoMS will them prompt the user to select a previously saved configuration. At this time, the HCoMS reads the saved configuration file and performs the automated configuration on the system. After this, the HCoMS overwrites the current runtime code location with the saved version of the runtime code. In addition, the HCoMS then instructs the user about the physical structure of the system and waits for acknowledgement from the user. At this stage, the system is successfully restored to the previously saved configuration. The user can now initiate a production run by launching the ordering system and placing product orders.

Chapter 4 Testing Methodology

4.1 Introduction - Overview of System Testing

This chapter is dedicated to the developed tests that need to be performed on the system to verify the system operation and subcomponents. The testing procedure is compiled in such a way that the fundamental components and functions are validated first, and then ultimately used in subsequent testing to validate the complete system in entirety. The sequence in which tests will be conducted can be better explained by referring to Figure 4.1. In Figure 4.1, notice that the dark grey coloured blocks represent the sections of the system that require verification in each test. In addition, the light grey coloured blocks denote general sections of the system that are required for the system to operate, and lastly, the white coloured blocks represent the sections of the system that have already been verified by prior testing.

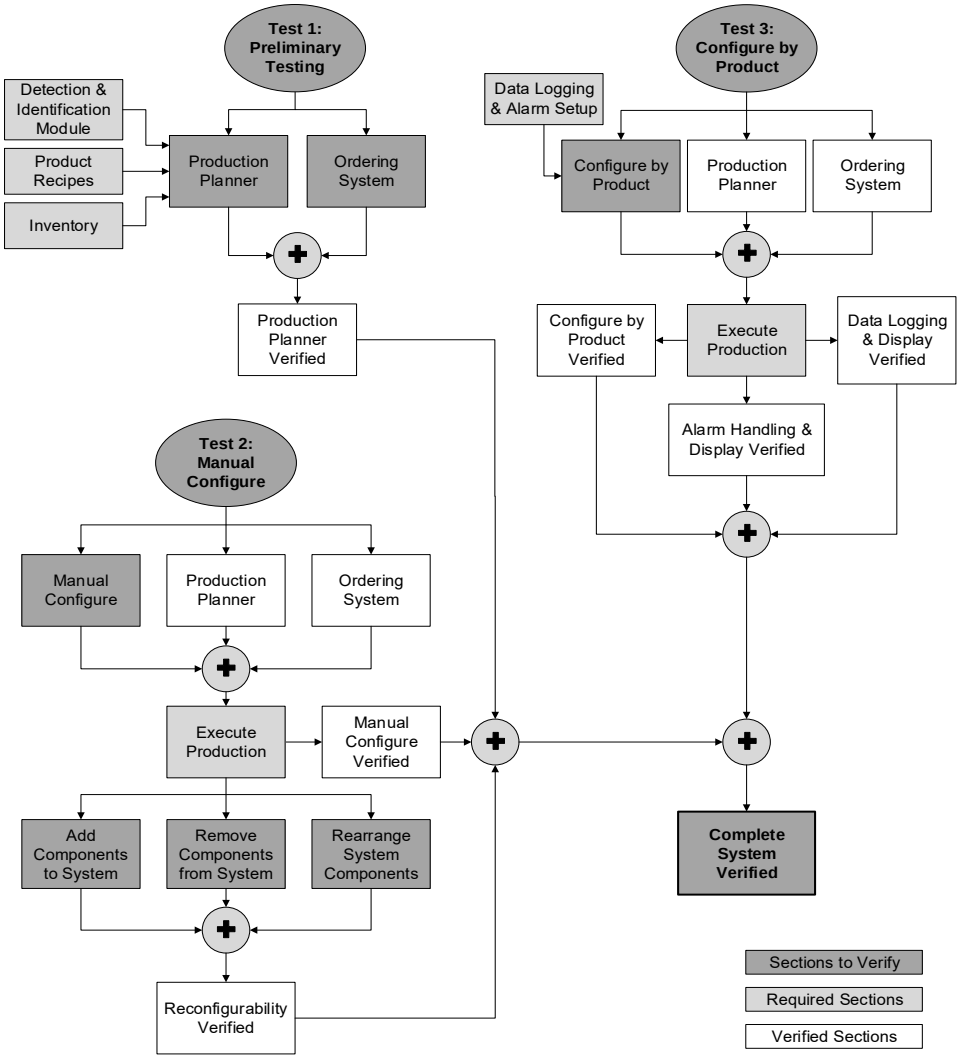


Figure 4.1 Flow diagram for verification testing

Initially, the first test is developed to verify all of the fundamental functions (the ordering system in conjunction with the production planner). Additionally, the detection and identification module is integrated with the production planner, where the production planner will access product recipes and the parts inventory from the database during the test. Subsequently, the second test will demonstrate the reconfigurability of the system. The “manual configure” mode will be utilized to initially configure the system and will demonstrate the virtual wiring ability that the system possess. During this test, it will be revealed how the system handles the adding, removal and rearranging of system hardware components. Afterwards, in contrast with the second test, the third test will utilize the “configure by product” mode of the system and verify the alarms and data logging capabilities of the system (SCADA capabilities). This test will reveal how alarm and data logging are set up; and eventually retrieved and handled by the user. Ultimately, the results from all the tests will verify the system as a whole.

4.2 Test 1: Preliminary Testing

This test is intended to validate all of the fundamental functions of the system, where all the validated components are going to be used in subsequent testing. The intention is to test only the functionality of the preliminary software components and to exclude the hardware components (as hardware devices will be verified in the subsequent sections). The preliminary testing is required to illustrate how the ordering system and auxiliary software components integrate with the production planner. The auxiliary software components under test include the detection and identification module, scheduler module (part of production planner) and the system configurator (generated system code). In addition to the auxiliary software, this test requires to validate access reads and writes to the parts inventory, as well as the product recipes and device capabilities databases.

Before the test can be performed, the following must be done beforehand. First of all, the products that the system will be able to build (products can be added to and removed from the system) must be specified and the respective product recipes must be included in the system database. In addition to product recipes, the inventory must contain all the parts that are needed to build the products as well as the adequate quantities of these. Furthermore, the device installers that enable the

system to identify each device and inform it about the capabilities of each device, must be included and run at least once. Additionally, the OPC server must already be set up to enable communication between the production planner and the required hardware devices (see support video in Appendix A). While only the software functionality of the system is verified during this test, it is still required to connect the hardware devices to the system network in order to test the software operation (testing the DIM).

With this in mind, the sequence in which the test will be performed, along with a summary of the expected results, can be seen in Table 4.1. The test is initiated by opening the ordering system to show the front panel, opening the DIM to monitor which devices are detected, and running the software. To clarify, the production planner is designed to run as a background process and thus do not need to be opened and will not show its front panel. After the software is started, it is expected that no devices should be shown in the devices connected window (DIM front panel) (no devices connected yet) and there should be an empty ordering screen that requires the user to place product orders. Next, the devices are powered and plugged into the system Ethernet network. At this stage, it is assumed that the devices plugged into the network will be detected and displayed in the DIM as it establishes connection with the OPC server (it should also update if the devices are removed from the network).

For the next step in the procedure, the user must place the desired batch order shown in Table 4.2. This is done by selecting a product from the drop-down menu, specifying a quantity and clicking add. It is assumed that the system will calculate if enough parts are available in the inventory; if the required devices needed to build the product are connected; and then update the schedule for the assembly process. After this, the user can repeatedly add, remove or rearrange orders to achieve the desired batch. For the purpose of this test, the procedure to order the desired batch will be done in the sequence given by Table 4.3. Each order will be placed in this sequence and then manipulated to achieve the final desired batch order. It is expected that the production planner will each time (after each order placement) compute if the updated orders can still be accomplished and also update the schedule in the desired sequence on the order screen. The last action to perform

will be that the user clicks “Order” to order the batch. The expected result is that the ordering system sends a message to the system configurator to initiate the configuration process.

Table 4.1 Test 1 sequence and expected result summary

	Action	Expected Result
1	Open ordering system	Shows ordering system front panel with no orders placed
2	Open DIM front panel	Shows DIM front panel with the devices currently connected to the network
3	Power and connect devices (PLCs) to the network	Identify devices as connected to system network
4	Place a blank order (no product and zero quantity)	Prompt to user that this is not allowed
5	Place an order with zero quantity	Prompt to user that this is not allowed
6	Place rest of product orders as shown in Table 4.3	HCoMS uses recipe to calculate if enough parts are available to build current product order and test if required devices are connected. Queues orders as it is added to the order screen, calculate and update production schedule on every order placed.
7	User rearrange and remove orders to obtain the desired batch order	Production planner determines if orders are still valid and order screen updates order queue and schedule
8	User orders batch by clicking “Order”	Ordering screen closes and sends instruction to HCoMS controller to start configuration

Table 4.2 Desired batch order

Product	Quantity
Product A	1
Product B	2
Product C	3

Table 4.3 Ordering sequence

Product	Quantity
Blank/Empty	0
Product C	0
Product C	3
Product B	2
Product A	2
Product A	1

4.3 Test 2: System Configurability

The objectives of this test is to demonstrate the ease in configuring the system using the “manual configure” function of the system, as well as demonstrate the reconfigurability of the system overall. Furthermore, the test must verify the “virtual wiring” ability of the system and the ability to generate (script) the code for the user to wire up. In addition, the test is intended to reveal how the system will automatically configure itself, based on information captured from the user during system prompts. The test must show how the system renders the front panel (user interface) based on the configuration. Furthermore, the test must reveal how the software handles change in the system when adding, removing and rearranging hardware components.

Prior to performing this test, the following must first be prepared. Firstly, the hardware devices to be used in this test must be confirmed. This is achieved by programming each device PLC with functional behaviour and developing device installers for each (and installing it). This test will be executed with three different hardware devices. To simplify explanation, each of these hardware devices will be given the functional behaviour of well-known components, namely multiplexers and de-multiplexers. To be more specific, one PLC (PLC 1) will have the functionality of a multiplexer, another the functionality of a de-multiplexer (PLC 2), and the last PLC will have the combined functionality of both and be able to dynamically change between it (PLC 3). To finalize the device installers for each, the VIs used in the block diagram (in the code) must be developed. An example of these can be seen

in Figure 4.2 (front panel) and Figure 4.3 (block diagram VI). Figure 4.3 shows the block diagram VIs for the multiplexer (at 1), de-multiplexer (at 2), and both instances of the dynamic polymorphic VI (at 3).

In addition, it is required that the CSV files for each device are created for setup on the OPC server, whereafter these devices are added to the OPC server (see support video in Appendix A). For the purpose of this test, the ordering system is hard coded (temporarily changed) to place a mock order using a mock product, to perform a process using the devices with its currently programmed behaviour (a mock process using multiplexers and de-multiplexers). In addition, the IO attributes for each device is also disabled (disable alarm and logging).

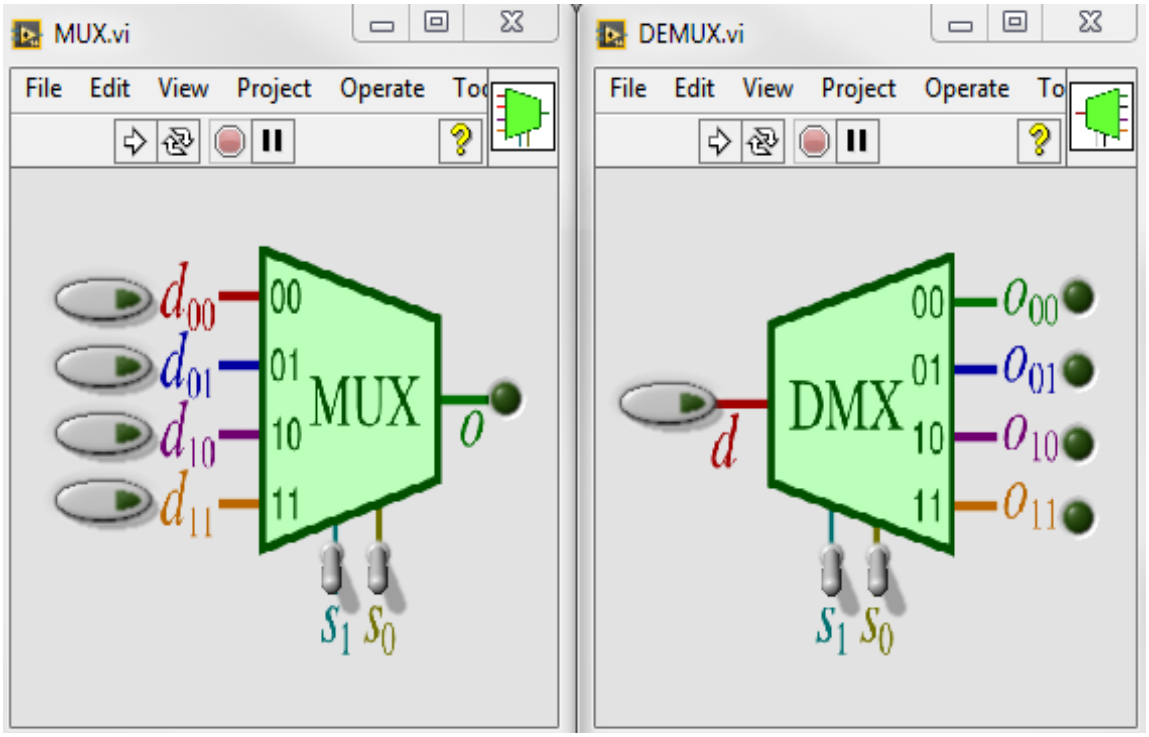


Figure 4.2 Front panels of devices

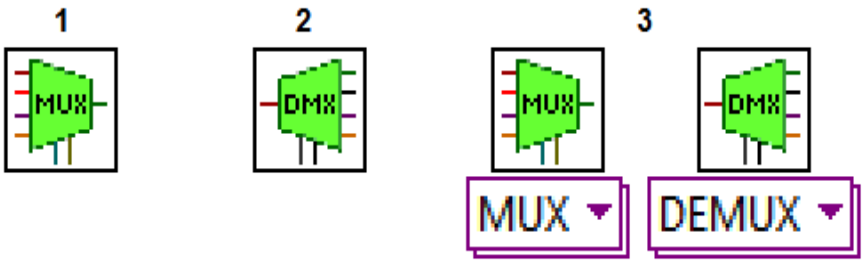


Figure 4.3 Device VIs used in block diagram code

At this point, the preliminary setup for the test is completed and it is required that the user resolve the system layout before starting the configuration process. The user can utilize the included (built in) help files to aid in this decision-making. For the purpose of this test, the chosen positions on the front panel for each device during each configuration is specified as shown in Table 4.4. In addition, the system will be repeatedly reconfigured during this test and the device positions in each case is shown in Table 4.5. With this in mind, the test can be commenced.

Table 4.4 Initial front panel setup

Device	Grid Positions	Orientation
Device 1	4	1
Device 2	5	1

Table 4.5 Position of devices during each test configuration

Testing	Position 4	Position 5
Initial Setup	PLC 1	PLC 2
Add and Remove	PLC 1	PLC 3
Rearrange and change polymorphic VI	PLC 3	PLC 2
Rearrange	PLC 2	PLC 3

For the procedure followed during this test along with a summary of the expected results, refer to Table 4.6. To initiate the test, the hardware devices must be powered and connected to the system Ethernet network. Next, the user can run the main application and choose the “manual configure” function from the main menu. At this stage it is assumed that the HCoMS controller will open the devices connected window (DIM front panel), detect the devices connected to the network and wait for user to select the devices to be used.

After this, the HCoMS controller will prompt the user about the orientation of each device. The user can now respond to these prompts by using the information provided in Table 4.4. At this time, it is expected that the system will guide the user through the whole configuration process like an install wizard (One prompt at a time in a required sequence). After all the information are acquired from the user and confirmed, it is expected that the HCoMS will configure the system devices

automatically. After the devices are configured, it is expected that the user will be presented with a popup window that contains system generated code (scripted) that is based on the current configuration. Here it is required that the user wires (virtual wiring) the VIs on the block diagram together to specify how the system devices interconnect with each other. After the device VIs are wired together, the user can save and close the runtime code and start the runtime process. At this stage, it is expected that the front panel will be rendered to reflect the current configuration, and that the system will operate according to how the system is virtually wired in the runtime code. Afterwards, the user can exit the runtime process to conclude the initial part of the test.

Table 4.6 Test 2 sequence and expected result summary

	Action	Expected Result
1	Power and connect devices (PLCs) to the network	User powers and connects devices to the network
2	Open and run HCoMS software	Show HCoMS main menu and allow user to choose from menu
3	User choose “Manual Configure” from menu	HCoMS opens DIM, detects devices and allow user to choose which to use
4	User selects devices to use	HCoMS prompts user about orientation and grid position of devices
5	User specified grid and orientation position as shown in Table 4.3	System starts automated configuration, configures devices, script runtime code and presents it to user
6	User connect and modify how devices are wired together, saves and exit runtime code	System configured, allow user to either choose and reconfigure system using other options (modes) or run current configuration (open runtime client)
7	User opens runtime client	HCoMS opens front panel and renders graphics based on user’s input from system prompts
8	User tests operation of devices displayed on front panel	Devices operate according to how the runtime code is wired
9	User stops the runtime client and repeatedly changes the configuration as shown in Table 4.3 by repeating steps 4 to 8	System updates the runtime front panel, and devices operate according to user modified runtime code
10	User exits HCoMS application	Application performs garbage collection and closes

At this stage, the user can repeatedly reconfigure the system by using the preceding steps to achieve each configuration shown in Table 4.5. Here it is expected that the user will perform each configuration; the system will render the updated front panel for each; and that the system operates according to the virtually wired runtime code in each case. After all the configurations are demonstrated, the test is concluded.

4.4 Test 3: Information Manager

The main aims of this test are to reveal how the system handles data logging and alarming, as well as how it retrieves and displays this data. In addition, the test must demonstrate how the system utilizes the “configure by product” function of the system to configure itself automatically, based on a chosen product. Before the test can commence, the following must be done beforehand to set up the test. Firstly, a mock product that requires the hardware devices to be monitored and logged, is created and added to the system. Thus, the system will configure itself, based on the mock product. In addition, the steps from Test 1 regarding product recipes, inventory and device capabilities must be updated and the configuration file with the recommended IO attributes (for alarming and logging) must be included. After the preliminary setup is completed, the test can get underway. The sequence in which the test will be performed, along with a summary of the expected results, can be seen in Table 4.7.

To commence the test, the required hardware devices must be powered and connected to the system network. In addition, the user can start the main application and choose “configure by product” from the main menu. At this stage, it is expected that the software will open the ordering system and wait for a response from the user. Here it is expected that the user places an order using the mock product that was added to the system. After ordering the specified mock product, it is anticipated that the system will configure itself, based on the product specifications automatically, which includes that it will generate and wire (script) the required internal code (VIs and connections). At this stage, the system will prompt the user with instructions, whereafter the user is required to arrange the physical layout of the system and acknowledge back to the system on completion. Because a mock product was used, the user can simply acknowledge without arranging the physical equipment. At this stage, the system is fully configured and the runtime process can

be executed. Furthermore, during the runtime process the following are expected: Firstly, the front panel is rendered identical to what was presented to the user. Secondly, real-time data changes and alarms are occurring on the front panel. Thirdly, the user can at any time enter either the historical trend or alarm windows, and retrieve historical data recorded during the runtime and service the system alarms that occurred. Once the prior activities are demonstrated, the test is finished.

Table 4.7 Test 3 sequence and expected result summary

	Action	Expected Result
1	Power and connect devices (PLCs) to the network	User powers and connects devices to the network
2	Open and run HCoMS software	Show HCoMS main menu and allows user to choose from menu
3	User choose “Configure by Product” from menu	HCoMS opens ordering system and waits for user to place order.
4	User places order by selecting a mock product	HCoMS opens the IO attributes, and allows the user to change them if desired
5	User saves IO attributes	System starts automated configuration, configures devices, scripts runtime code and prompts the user about the physical layout of devices
6	User acknowledges that placement is complete	HCoMS allows the user to either choose and reconfigure system using other options (modes) or run current configuration (open runtime client)
7	User opens runtime client	HCoMS determines the placement of the devices, open and renders front panel graphics as such
8	User observes the operation of devices displayed on front panel	Devices operate according to recommended configurations
9	Any time that device alarms occur	User is able to acknowledge alarms using alarms detailed view window
10	Any time during operation	User is able to access and retrieve logged device data using the historical trend

Chapter 5 Results and Analysis

5.1 Introduction

This chapter scrutinizes the results obtained from the tests performed in Chapter 4. It discusses outcomes that were expected; where the outcomes differed; how these were mended; as well as what is lacking in the system.

5.2 Test 1: Preliminary Testing

The purpose of the test done in section 4.2 was to test the functionality of the fundamental software components. This includes how the DIM, ordering system and production planner function together. The procedure of how the test was done along with both the expected and obtained results are compiled and can be seen in Table 5.1.

The software components (DIM and ordering system) were opened and the test commenced. The success of the DIM (Table 5.1 at 2) is shown in Figure 5.1. As expected, the DIM detects devices as the devices are being connected to the system network. In addition, the DIM also removed devices from the display when devices were disconnected from the network. How the software detects and identifies devices and displays them on the screen can be seen in the supporting video evidence in Appendix B1.

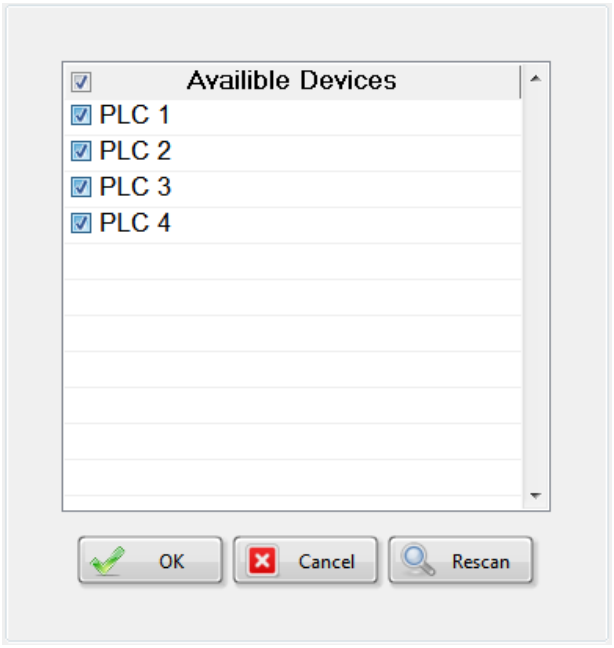


Figure 5.1 Devices detected on the system network

Table 5.1 Test 1 sequence of actions and summary of results

	Action	Expected Result	Obtained Result
1	Open ordering system	Shows ordering system front panel with no orders placed	As expected
2	Open DIM front panel	Shows DIM front panel with the devices currently connected to the network	As expected
3	Power and connect devices (PLCs) to the network	Identify devices as connected to system network	As expected
4	Place a blank order (no product and zero quantity)	Prompt to user that this is not allowed	As expected
5	Place an order with zero quantity	Prompt to user that this is not allowed	As expected
6	Place rest of product orders as shown in Table 4.3	HCoMS uses recipe to calculate if enough parts are available to build current product order and tests if required devices are connected. Queues orders as it is added to the order screen, calculate and update production schedule on every order placed.	HCoMS uses recipe to calculate if enough parts are available to build product order and tests if required devices are connected. Queues each new order at the front of the production queue as it is added to the order screen, calculates and updates production schedule on every order placed.
7	User rearrange and remove orders to obtain the desired batch order	Production planner determines if orders are still valid and order screen updates order queue and schedule	As expected
8	User orders batch by clicking “Order”	Ordering screen closes and sends instruction to HCoMS controller to start configuration	As expected

Furthermore, the ordering system and the production planner also delivered satisfactory results. The sequence in which the product orders were placed is given in Table 4.3 in Chapter 4. Afterwards, these orders were manipulated to achieve the desired batch order as shown in Figure 5.2 (see Table 4.2). The results obtained from this part of the test are also shown in the supporting video evidence in Appendix B2 and are as follows: The first product order that was placed was a blank order (order with no product or quantity specified) and as expected, the system

prompted the user that it is not allowed. The second attempt was an order with a product specified, but with zero quantity, and as expected, the system again prompted that the order is not allowed.

The remainder of the orders shown in Table 4.3 met the requirements for valid orders and was placed one at a time. The first valid order (third attempt) was placed and the results show that an order is added in the order screen. This shows that the production planner executing in the background has successfully accessed information from the parts inventory, product recipes and device capability databases respectively, and determined if the order can be placed. Here the system will access the product recipe to obtain information about the parts required to assemble the product. Next, this information is compared to the parts inventory to determine if the required parts are available and if sufficient quantities thereof are in stock. If this is met, the production planner will test which devices are connected to the system network, access the device capabilities for each one that is connected, and compare it to the information from the product recipe to determine if the required devices to assemble the product are available. If all these requirements are met, the order is valid, and can be scheduled and added to the order queue in the ordering system. The production planner schedules the start time for the batch order, to commence ten minutes from the current system time.

The ordering system displays which product is going to be build, the quantity thereof, the estimated duration of the process, and the estimated date and time of completion. Furthermore, the results show how the ordering screen updates when new orders are added or if orders are removed or moved up or down the order queue. The production planner each time determines if all the current orders are valid to be placed, and sends this information to the ordering system to refresh the ordering screen with the updated schedule. After all the orders are placed and manipulated to reflect the desired batch order (Figure 5.2 and Table 4.2), the user submits the order by clicking “Order”, and the system responds by starting to configure itself automatically (by sending the instruction message in the background). This can also be seen in the supporting video evidence in Appendix B2.

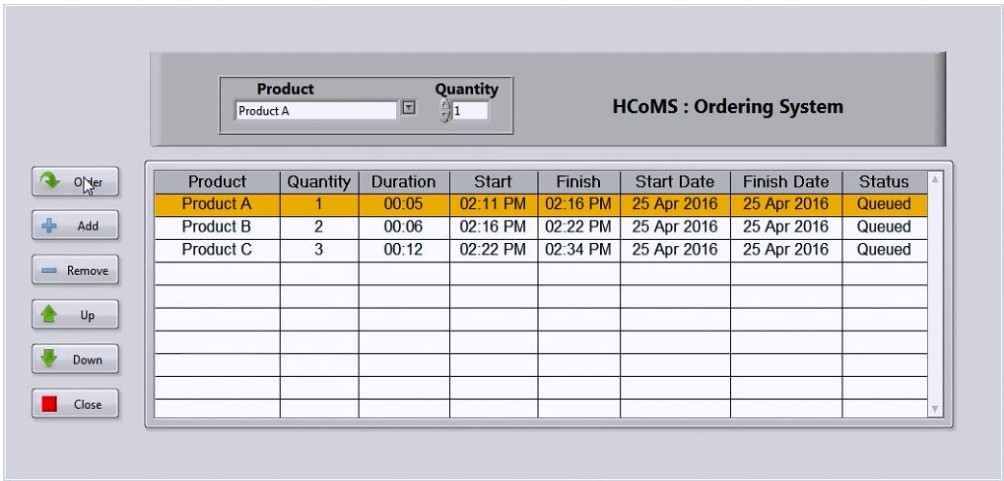


Figure 5.2 Desired batch order result

The system successfully acquired the desired batch order shown in Figure 5.2 (Table 4.2) from the user's input. However, the way that the system scheduled and queued the orders were undesired (unexpected results, see Table 5.1 at 6). Each new order that was placed enters at the front of the order queue, thus delaying the existing orders to a later start time (the last order is first in the queue). This was investigated and it was found that incorrect priority was assigned during the programming stages (Figure 5.3). As the ordering system gathered information to send to the production planner, the information was provided with the new order appended at the front (of the array) of the list of orders. This was rectified by correcting the source code from what is seen in Figure 5.3 at (1) to what is shown in Figure 5.3 at (2). After the program was updated, the test was repeated and the desired result was obtained, where the ordering system added new orders to the back of the order queue. This can be seen in the supporting video evidence in Appendix B3.

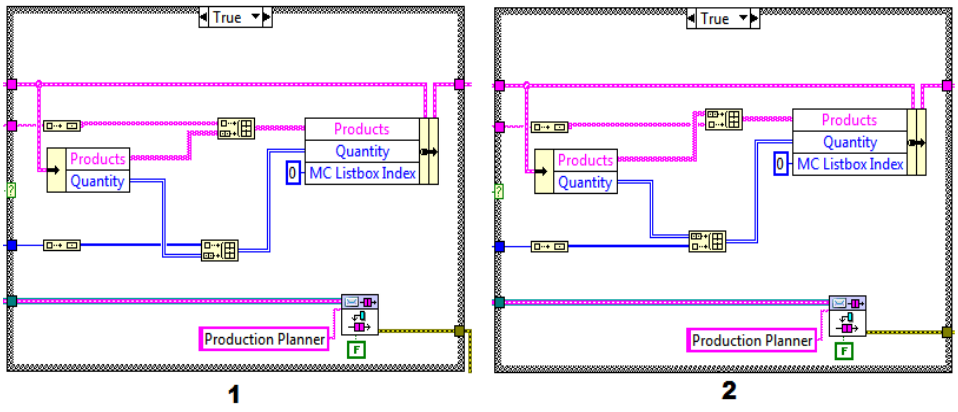


Figure 5.3 Code corrected

5.3 Test 2: System Configurability

The purpose of Test 2 (as was stated in section 4.3) was to demonstrate the ease in configuring the system using the “manual configure” function. In addition, the test had to show the ability of the system to automatically generate functional code (using scripting) and how the system uses the concept of virtual wiring to allow a user to interconnect the system hardware components together. Furthermore, the test had to show how the system will guide a user through the configuration process by using sequential system prompts and automatically configure itself once it acquired all the required information from the user. The test also had to illustrate how the runtime front panel is rendered based on user input, as well as how the system handles change when the system is reconfigured by adding, removing or rearranging system components. The procedure of how Test 2 was done along with the comparison between the expected and obtained results are compiled and shown in Table 5.2.

The results obtained from Test 2 can be better explained by also referring to the support video in Appendix C. The test was initiated and the user was able to select the “manual configure” function of the system. This resulted in the DIM opening and prompting the user to choose from the available PLCs connected to the system network. This means that the DIM successfully detected, identified and presented the devices connected to the network. Next the user was prompted to choose (from only the selected devices) in which orientations and grid positions the devices must be displayed in on the front panel. How the user selects the initial setup is shown in Table 4.5. The results show how the system starts to automatically configure once all the information required from the user are captured. This shows how the system (in the background) creates the required variables, maps them to the items on the OPC server, and maps it to their respective front panel objects.

Table 5.2 Test 2 sequence of actions and summary of results

	Action	Expected Result	Obtained Result
1	Power and connect devices (PLCs) to the network	User powers and connects devices to the network	As expected
2	Open and run HCoMS software	Show HCoMS main menu and allow user to choose from menu	As expected
3	User chooses “Manual Configure” from menu	HCoMS opens DIM, detects devices and allows user to choose which to use	As expected
4	User selects devices to use	HCoMS prompts user about orientation and grid position of devices	As expected
5	User specified grid and orientation position as shown in Table 4.3	System starts automated configuration, configures devices, script runtime code and presents it to user	As expected
6	User connects and modifies how devices are wired together, saves and exits runtime code	System configured, allows user to either choose and reconfigure system using other options (modes) or run current configuration (open runtime client)	As expected
7	User opens runtime client	HCoMS opens front panel and renders graphics based on user’s input from system prompts	As expected
8	User tests operation of devices displayed on front panel	Devices operate according to how the runtime code is wired	As expected
9	User stops the runtime client and repeatedly change the configuration as shown in Table 4.3 by repeating steps 4 to 8	System updates the runtime front panel, and devices operate according to user modified runtime code	Results were as expected, system successfully reconfigured each time, updated runtime front panel and operated according to user modified runtime code
10	User exits HCoMS application	Application performs garbage collection and closes	As expected

Directly after the automatic configuration had been completed, the system opens the runtime code and allows the user to virtually wire the devices as desired. The supporting video evidence in Appendix C shows how “generated code” is present in the screen and how the user can move the elements around and connect them.

When the wiring is done, the user saves and exits the runtime code window. The next result shows the user running the client front panel. The result is that the front panel is rendered with the devices in the positions as given by Table 4.5. Examples of the rendered front panels is shown in Figure 5.4 and Figure 5.5. Here the user operates the system, revealing that the components used are “virtually connected” and operates satisfactory. After this, it is shown how the user exits the runtime front panel and reconfigures the system with the next configuration shown in Table 4.5 by repeating the procedure followed during the initial setup. Additionally, the video clip in Appendix C shows how the user changes the functionality of the dynamic module (PLC 3) during the test. This proves that a dynamic module can perform the functions of multiple single function modules. At this stage of the test, the user repeatedly stops the runtime process, follows the system prompts to setup the next configuration in Table 4.5, modifies the current runtime code, and restarts the runtime process. This result shows that the user is able to add, remove and rearrange the modules (VIs) and rewire as required. Examples of the user modified code during each test configuration are shown in Figure 5.6. Most importantly, this test demonstrates the reconfigurability of the system and the ease of system setup.

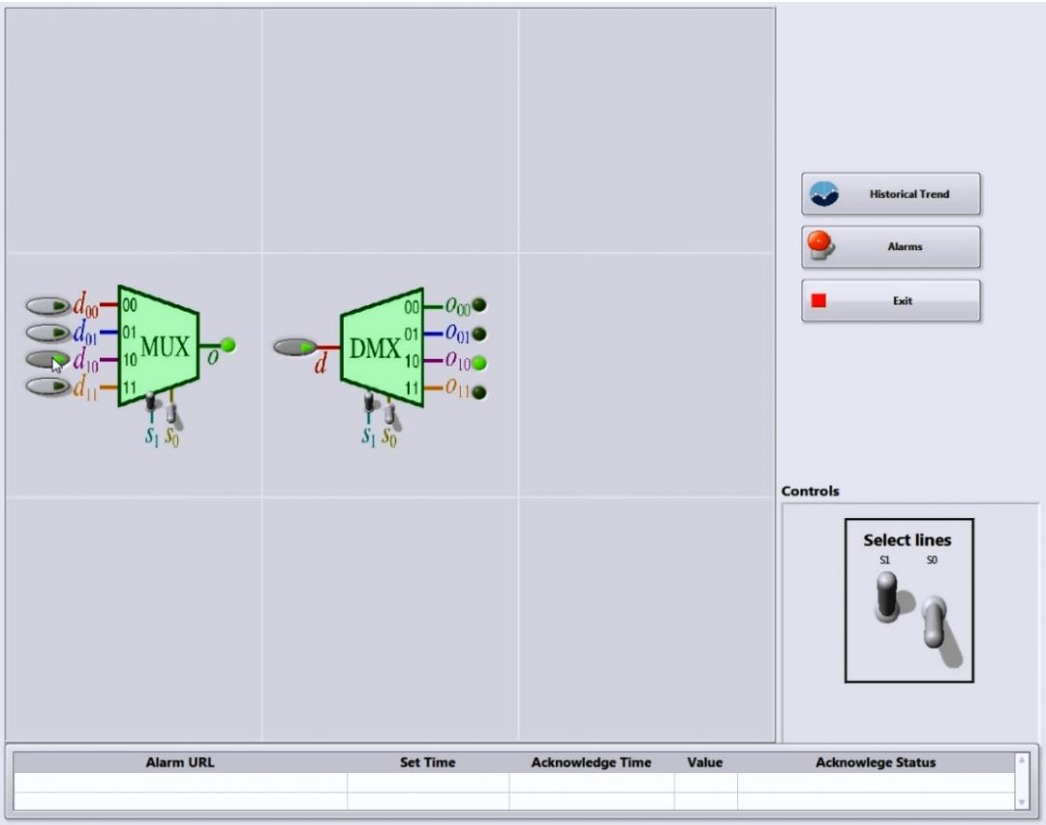


Figure 5.4 Front panel during initial setup configuration

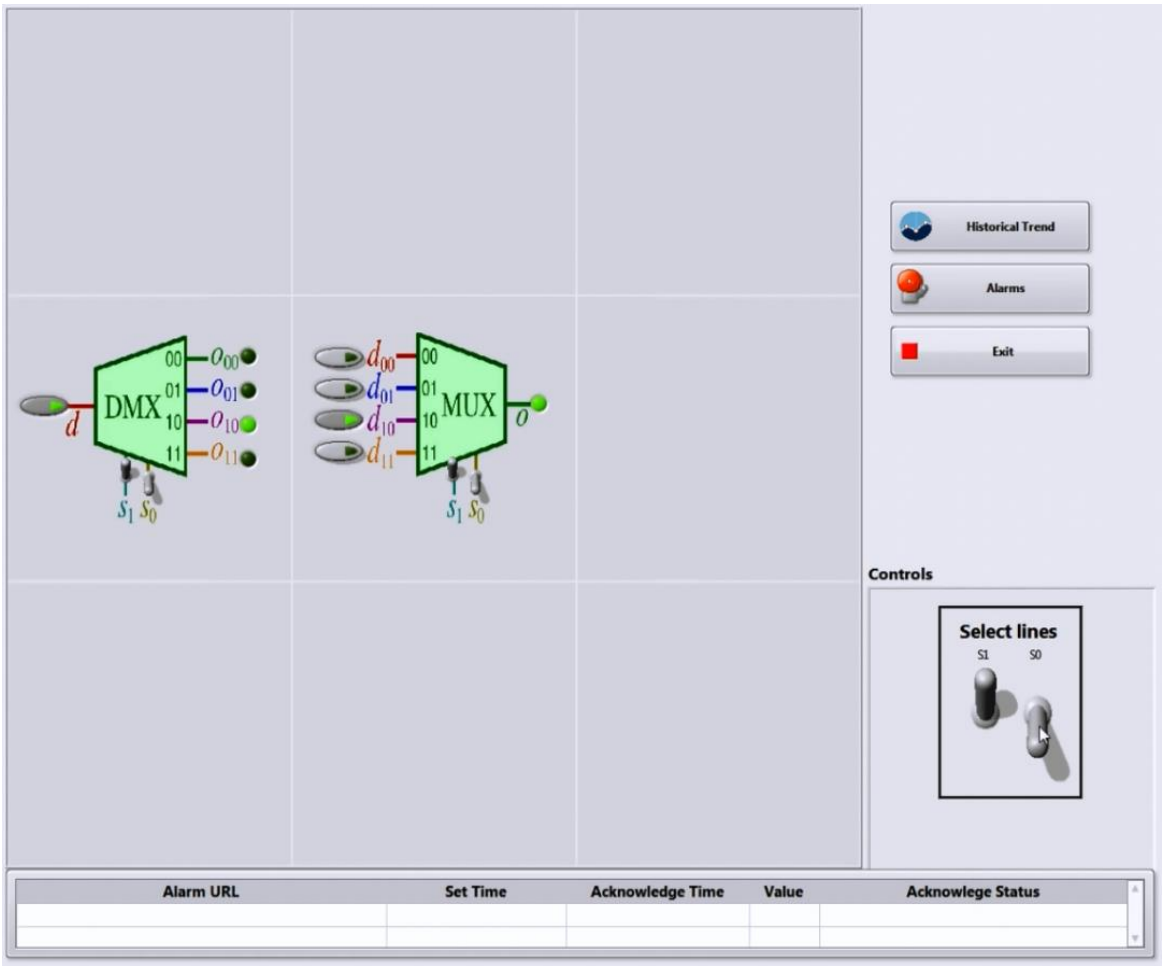


Figure 5.5 Front panel during final setup configuration

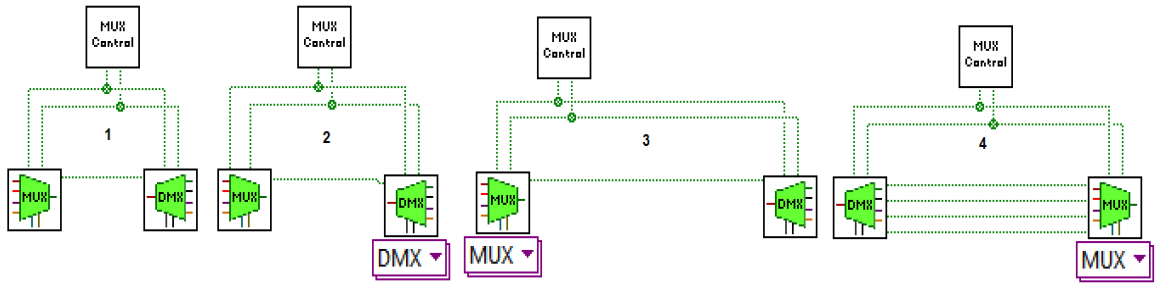


Figure 5.6 Runtime code during each configuration test

5.4 Test 3: Information Manager

The purpose of Test 3 (as was stated in section 4.4) was to establish how the system automatically configures without user input based on product requirements by using the “configure by product” function. In addition, the test had to show the capability of the system to reveal real time data to the user (on the front panel), warn and alarm about undesired and/or unsafe conditions, and how these warnings and alarms are

treated (handled). Furthermore, the test had to reveal how a user would retrieve logged information using the (built-in) historical trend, manipulate it and display it on the screen. For the procedure of how Test 3 was done and the comparison between the expected and obtained results, refer to Table 5.3. In addition, also refer to the support video in Appendix D to aid in the discussion of the results following below.

Table 5.3 Test 3 sequence of actions and summary of results

	Action	Expected Result	Obtained Result
1	Power and connect devices (PLCs) to the network	User powers and connects devices to the network	As expected
2	Open and run HCoMS software	Show HCoMS main menu and allow user to choose from menu	As expected
3	User choose “Configure by Product” from menu	HCoMS opens ordering system and waits for user to place order.	As expected
4	User places order by selecting a mock product	HCoMS opens the IO attributes, and allows the user to change them if desired	As expected
5	User saves IO attributes	System starts automated configuration, configures devices, scripts runtime code and prompts the user about the physical layout of devices	As expected
6	User acknowledges that placement is complete	HCoMS allows the user to either choose and reconfigure system using other options (modes) or run current configuration (open runtime client)	As expected
7	User opens runtime client	HCoMS determines the placement of the devices, open and render front panel graphics as such	As expected
8	User observes the operation of devices displayed on front panel	Devices operate according to recommended configurations	As expected
9	Any time that device alarms occur	User is able to acknowledge alarms using alarms detailed view window	As expected
10	Any time during operation	User is able to access and retrieve logged device data using the historical trend	As expected

The results from Appendix D show the user choosing the “configure by product” function and then being presented with an ordering screen. Notice that this ordering screen is different from to the ordering screen from Test 1; the reason being that for this method the user only has to provide a product and its quantity. The user places the mock order (order tailored for this test) and is then presented with a popup window to modify the IO attributes of the connected devices. This shows that the system successfully detected that devices that are able to assemble the selected (mock) product are connected to the network; and the system successfully determined that enough parts are available in stock to assemble the product. The user can now perform the desired changes to the IO attributes of the devices connected to the network and afterwards continue with “Save” (Figure 5.7).

Otherwise, if the default settings are desired, the user can just continue with “Save”. This shows how the system allows the user to change IO attributes, and enables the alarm and logging properties of the devices. After the user is done, the system starts to automatically configure. Note that no wiring screen was presented to the user. This shows that the system generated the functional code and wired it based on the product requirements (in the background). Next, the system prompts instructions to the user about the mandatory physical placement of the system devices by presenting the screen shown in Figure 5.8. Since a mock product was used, the user acknowledges with done. Now the user only has to start the runtime process by opening the runtime front panel. After the user opens the runtime front panel, the front panel screen is rendered with the devices in the same grid positions as stipulated by the instructions given to the user. This shows that the system automatically determined where the devices should be placed without requiring input from the user.

In addition, further results show how the system displays ongoing value changes in each device and also alarming on undesired values. This reveals that the system is able is to display real-time device data to the user. An example of this is shown in Figure 5.9. It is shown that alarms can be handled and acknowledged during any time of the runtime process. This shows that the system allows the user to “silence” and acknowledge alarms, as well as interact with the system based on these alarms.

The next result revealed during Test 3 was how a user can retrieve logged data using the system's historical trend window. An example of the historical trend data is shown in Figure 5.10. This shows that the system provides a function to specify and retrieve desired information, without "mining" or filtering through raw data in the database.

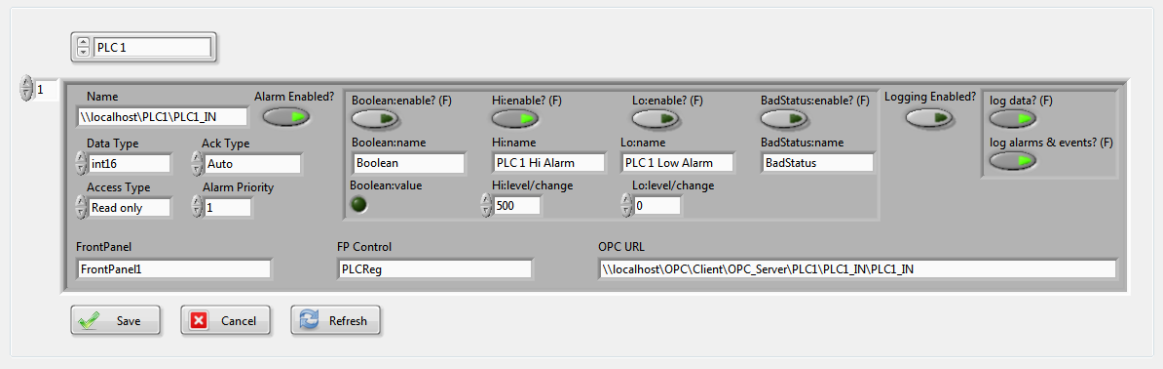


Figure 5.7 IO attributes modification screen

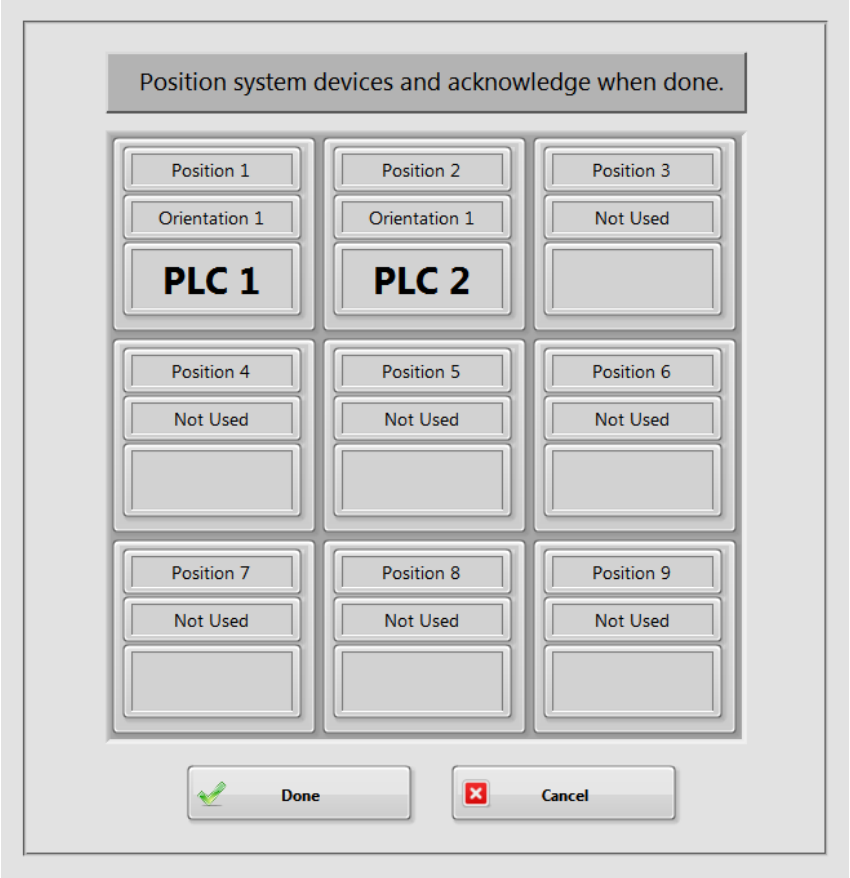


Figure 5.8 Instructions prompted to user

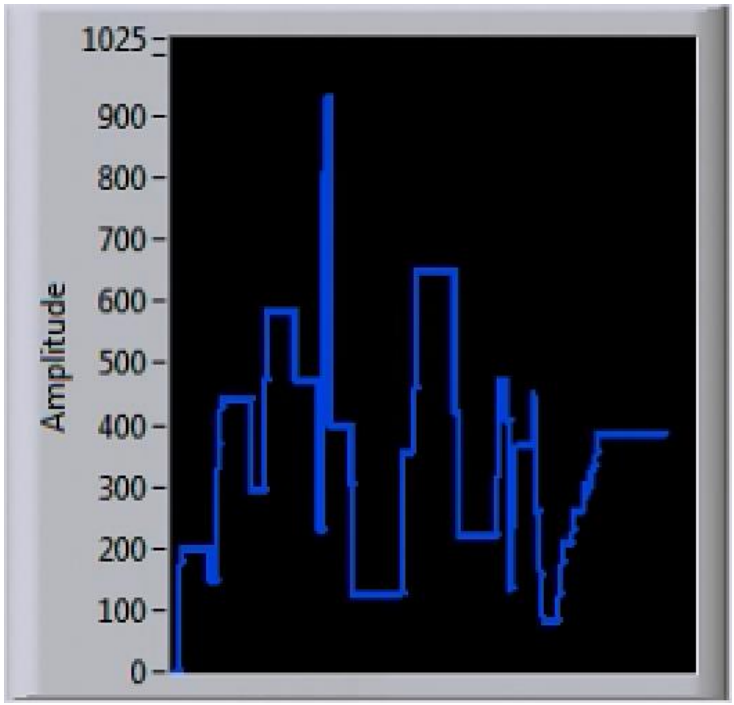


Figure 5.9 Real time device data displayed on front panel

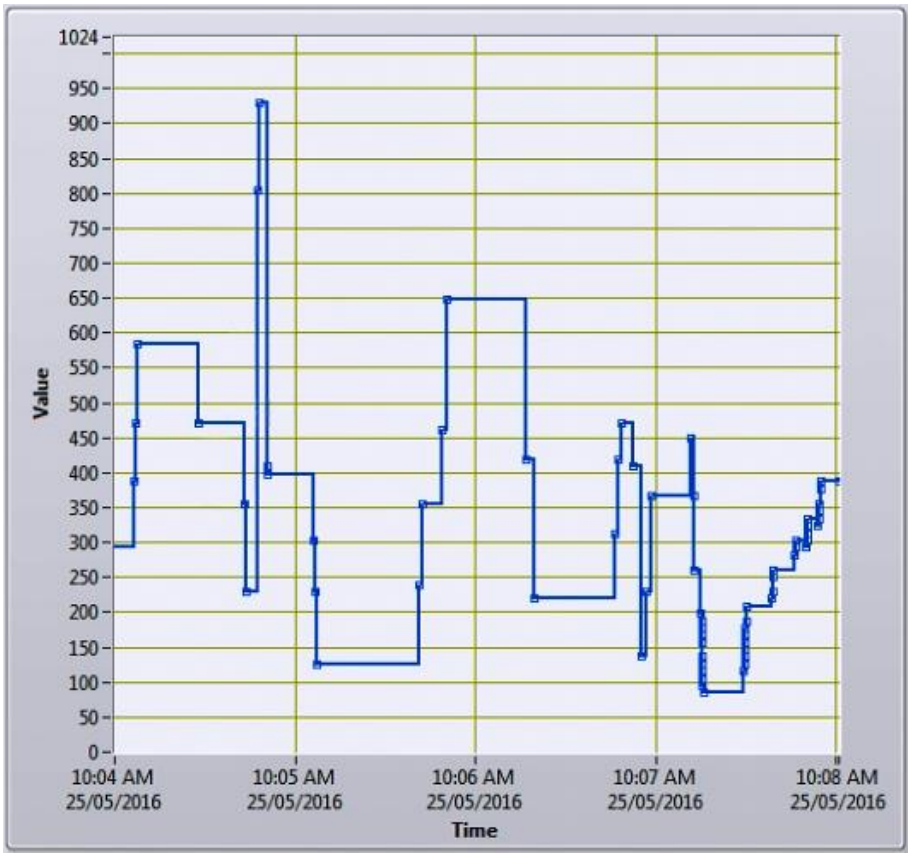


Figure 5.10 Data retrieved using Historical Trend

Additionally, also revealed from this test, is how a user would natively retrieve “raw” data from the Citadel database shown in Figure 5.11. Though it is undesired to retrieve data in this manner, it is still done during the test to compare the real-time data to historical trend data to raw database data. This can be seen in the supporting video evidence in Appendix D.

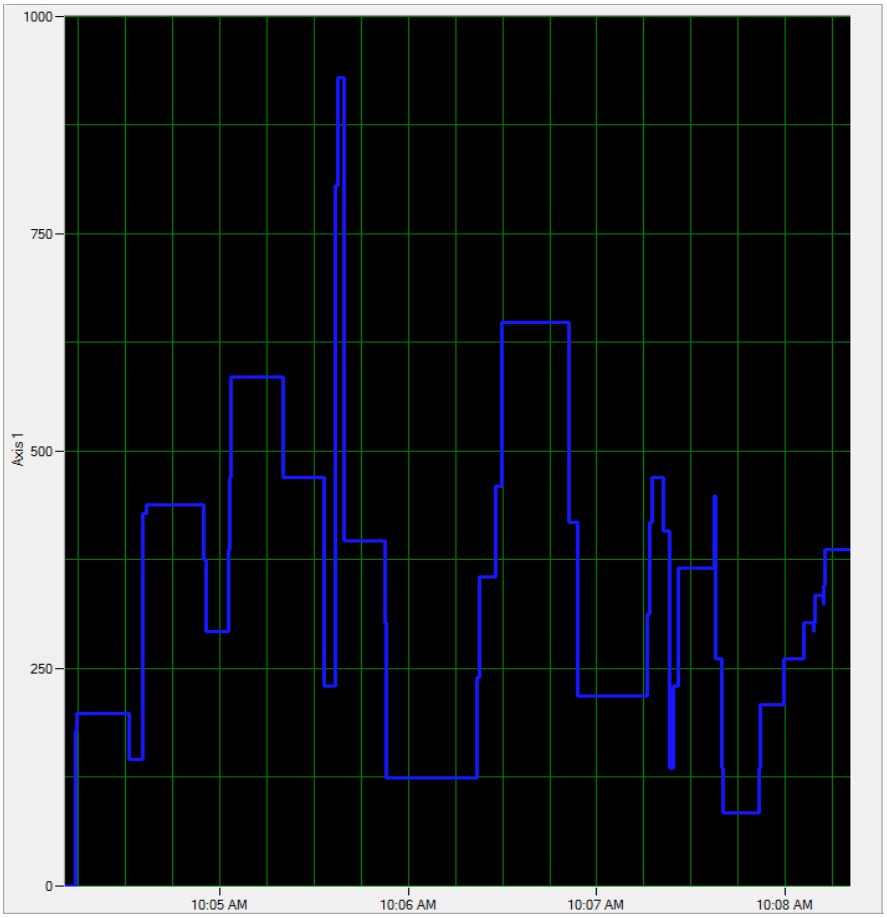


Figure 5.11 Raw data in Citadel database

5.5 Analysis and Summary of Results

This chapter revealed the results that were obtained during testing to validate all the capabilities and operations of the system. It discussed the expected outcomes, where these differed from what was obtained, and how these were mended.

The intention of the first test was to validate the functionality of the preliminary software components which included the DIM, ordering system and the production planner. The intention of the second test was to demonstrate the reconfigurability of the system by utilizing its “manual configure” mode. This test had to validate the “virtual wiring” ability of the system; the ability to generate runtime code; how the

system automatically configures based on user input; and how the system renders the front panel based on this configuration. The aim of the third test was to reveal how the system handles data logging and alarming, and how the system displays and retrieves this data. In addition, this test had to be done by utilizing the “configure by product” mode of the system.

To summarize, the results from the testing revealed that the system functions in accordance with the project objectives. The DIM, ordering system and production planner functioned successfully during all the phases of testing. This proves that, together, the fundamental software functioned satisfactory and shows that the system can execute planning and scheduling capabilities. In addition, the use of the DIM in conjunction with the device installers showed the ability of the system to add any new compliant device to the system.

Furthermore, the results show the satisfactory operation of each configuration mode where the user either has the final decision in configuring the system (manual configure), or allowing the system to configure itself and then following the instructions from the system prompts (configure by product). Moreover, the flexibility of the system is demonstrated by its ability to automatically generate runtime code. Depending on the configuration mode that was used, the system either generated the code (already wired), or only populated the base code and allowed a user to interconnect system devices as desired using the “virtual wiring” ability of the system. The “virtual wiring” ability of the system allows a user to initially interconnect system devices and correct existing code by changing connections between devices. Furthermore, the results have shown the success of the automated configuration routine. The automated configuration saves much time during the configuration stage when compared to a user setting up the system, and avoids the errors that could have been made by a user (human errors).

Finally, the results have also shown how the system rendered the front panel based on a configuration, how real-time process and alarm information was displayed on the front panel, and how this information was retrieved and displayed if/when it was logged. Overall, the results obtained were desirable. It is evident from the results that the development of the system successfully accomplished all that was specified in the project objectives.

Chapter 6 Contributions and Conclusion

6.1 Introduction

This is a closing chapter that recapitulates the project, revisits the research goals and objectives, expresses the contributions made, identifies future work to be done, and ultimately draws a conclusion to the research.

6.2 Summary

Chapter 1 introduced the project and placed it in perspective by stating the research problem at hand, identifying the hypothesis, articulating the research methodology and finally listing the objectives of the project. Chapter 2 contains a literature review of the field of study which included various aspects of reconfigurable assembly systems, control and monitoring issues of these, GUI design considerations, and planning and scheduling systems. Later, Chapter 3 documented the methodologies undertaken to develop the system. It specified the system hardware and software components utilized, how these function, and how these components are integrated. Next, Chapter 4 identified the tests and procedures that were developed to validate the system. And ultimately, Chapter 5 provided an analysis and discussion on the findings from the tests performed.

6.3 Research Goals and Objectives

The main objective of the study was to develop a hybrid control and monitoring system to be utilized with reconfigurable assembly systems. The monitor and control system had to either adapt the system configuration based on product specifications or allow a user to manually configure it as desired. In addition, the system also had to ensure the quality of the products assembled and provide this production information back to the user. Furthermore, the user interface of the system had to conform to high, internationally practised standards and be compatible with touch panels, tablets and smart phones.

These objectives were accomplished by firstly building the hardware devices to be used in the system; and then developing the software modules that are collectively required to construct the entire system, by using the methods which were discussed in Chapter 3. In addition, Chapter 3 also discussed the intended operation of the

complete system. Afterwards, tests were established in Chapter 4 and were set up in such a way as to verify each separate function (mode) of the system, as well as verify the major components of the system. After the tests were completed and results were obtained, these results were analysed and discussed in Chapter 5. This chapter revealed satisfactory results regarding the operation of the system and highlighted areas that can be looked at in future research. Most importantly, the results in Chapter 5 verified the operation of the system and ultimately prove the concept concerning a hybrid control and monitoring system developed for reconfigurable assembly systems.

6.4 Contributions

The project delivered a hybrid control and monitoring system with the following new implementations, configurations and contributions:

6.4.1 HCoMS System

The HCoMS system overall mainly consists of two parts, namely the graphical user interface and the background control software and integrated software modules. The HCoMS controller application was developed to be a multi-window application with multi-treaded concurrent processes in the background. The GUI screens used are designed to be intuitive and self-explanatory. The various integrated software modules are implemented in the system to add to the reconfigurability of the system and contribute towards easy system configuration and real-time operation. The HCoMS controller was developed to be compatible with devices from various different vendors through the use of OPC and to easily introduce and integrate new devices. This provides the foundation for the flexibility of the monitor and control system.

6.4.2 Device Installers

The device installers represent a vital component of the system that makes the system extremely integrable and were implemented to enable the HCoMS to utilize various compliant devices. These device installers were developed to automatically copy the files required for respective devices to the vital file locations in the system, and amend configuration files currently used in the system. As part of the project, various of these device installers have been developed for the assembly devices

used in the RGEMS laboratory, as well as the devices used during the testing phase of the project. In addition, code templates were included with the project files to enable current or future students to develop their own device installers for devices to be added to the system. These templates consist of the fundamental code structure and contain comments at strategic locations in the code that provide the developer with clear instructions on how to modify these code sections. Furthermore, the code comments are written in such a way that a student/developer can use the search text function to find all the locations in the code that require modification. As a result, current or future developers are able to easily add assembly devices to the system with minimal effort and error.

6.4.3 Detection and Identification Module (DIM)

The detection and identification module was developed and implemented in the HCoMS. By utilizing device installers, the DIM enables the HCoMS to scan the network and identify each device connected to it. In addition, the DIM was developed to continuously test if a connection is still present during runtime and convey this connection status back to the HCoMS controller.

6.4.4 Ordering System

An ordering system was developed to enable a user to place production orders during runtime. The ordering system is developed as a dynamic stand-alone module and is able to adapt based on system requirements. In addition, it integrates with the production planner to deliver a production schedule back to the user. Furthermore, it is designed with the ability to be used either on the same machine as an HCoMS, or another remote machine.

6.4.5 Production Planner and Scheduler

A production planner and scheduler was developed to provide the system with functions similar to the ERP and APS systems discussed in Chapter 2. The production planner consists of multiple compare and decide routines to enable it to plan and schedule product orders received from the ordering system. The production planner is developed to be a dynamic module that can easily be adapted through the use of device installers (from planning point of view). Essentially, the

production planner is designed to determine “what” is built “where,” “in which order,” and “with which resources”.

6.4.6 System Configurator

To aid with system configuration, the system configurator was developed and utilized in the HCoMS. The system configurator utilizes a dynamic state machine that can adapt its behaviour, an automated configuration routine and a code scripting algorithm which generates the runtime system code. Essentially, the system configurator utilizes the developed dynamic state machine to capture information from a user through a series of system prompts. In addition, the automated configuration routine was developed to perform repetitive setup actions for each of the devices connected to the system. These setup actions would take a user a few hours to days to perform on a large system (20 devices or more), where this routine will perform the same work in seconds to minutes with a guarantee that no human errors occurred. After the system is configured, it is required that the system runtime code is generated. To achieve this, a dynamic algorithm was developed to generate the runtime code by using VI scripting. As a whole, the system configurator was intended to guide a user through the entire configuration process.

6.4.7 Production Handler

The production handler was developed to manage the background control processes during runtime. Its intention was to instruct assembly devices to adapt their behaviour based on the requirements of the production run, and keep track of the status of the production process. In addition, the production handler was developed to render the runtime front panel based on configuration information obtained from the user. This was to provide the user with a geographical overview of the complete system.

6.4.8 Information Manager

The information manager was developed to display real-time production data on the system front panel and also display and handle system alarm conditions. Historical data are normally accessed natively through the Citadel database and/or the distribution manager. The information manager was developed to access, retrieve

and display this data. Furthermore, the information manager was developed to make it easy for a user to filter and browse through database data (in that it hides any irrelevant data from the user). To clarify the need for the information manager, the native methods to retrieve data from the Citadel database will also include data from other LabVIEW projects not related to the system.

6.4.9 System Assembly SMART Devices

An introductory physical version of an HCoMS has been built in the RGEMS research laboratory. This is an introductory version because new devices can be integrated and added in future. Each assembly device was assembled with an identical architecture from various subcomponents. Currently, these devices are used for the project, and due to their flexibility and modularity, are simultaneously used in other separate projects as well. This demonstrates the fact that the assembly devices are reconfigurable and easily interchangeable. As a result, current and future RGEMS students can benefit from having a range of readily available assembly devices to complete their projects, and if desired, also integrate newly built projects with an HCoMS.

6.5 Future Work

This subsection identifies what is worth investigating in the future. It states some recommendations and possible enhancements for future use.

6.5.1 Device Installers

The way that device installers are structured and implemented in the current system functions satisfactorily for the existing system. However, it is worth investigating different methods to improve these device installers. For example, these device installers might be developed to reside on each assembly device itself and install on an HCoMS when it is connected to the device network, similarly to how USB devices identify and install drivers. This will ensure that the initial running of the device installers is automated and not the responsibility of the user. In addition, this will make using an HCoMS easier with even less user intervention.

6.5.2 Ordering System and Production Planner

The ordering system and production planner together currently perform an excellent job in the system. However, the system can presently schedule only one series of

product orders and not parallel concurrent product orders. An improvement would be if the ordering system could handle multiple concurrent orders and if the production planner were able to schedule multiple resources to assemble these products concurrently. This recommended improvement will enable the system to schedule concomitant product orders, making the system run more efficiently by increasing productivity and reducing assembly device idle times.

6.5.3 Front Panel Rendering

At present, if a user encounters that the runtime front panel is erroneously setup or rendered, the user must exit the front panel and repeat the setup prompts to rectify this. An improvement would be if the user can simply change the position of the devices on the front panel during runtime, without repeating the setup steps. A recommendation is that the user can utilize a dropdown menu to select each device, or maybe drag and drop capabilities.

6.5.4 Runtime Code Scripting

The operation of the code scripting algorithm to generate runtime system code exceeds expectations. Currently, when the system has to determine the runtime code, it selects the first number of assembly devices available to be used and generates the runtime code accordingly. An improvement to this algorithm would be if a user can still have the final decision on which devices to use. For example, if the system selects to use devices 1 and 2 for a specific product, the user would be able to choose devices 3 and 4 to assemble the same product. This will aid in situations where certain devices require maintenance or are needed for other usage.

6.6 Conclusion

Reconfigurable assembly systems have ascended in the SA manufacturing environment due to the uncertainty of global markets. The majority of SA manufacturing exports are aimed at niche markets, which involves high varieties of products in small quantities. Due to this, SA manufacturing companies pursue the utilization of RAS to deliver a flexible platform that accommodates the required diversity in product manufacturing. RASs are complex flexible systems and need to be expertly monitored and controlled to operate at high efficiency. The systems that supervise and control RASs, must be just as flexible as the RAS itself. With this in

mind, this research study contributes towards the development of an HCoMS that intelligently integrates with and supervises RASs.

The utilization of a system such as an HCoMS enhances the overall functionality of a RAS. In essence, the HCoMS ensures that any compliant assembly device can be added to a RAS with ease. In addition, the HCoMS guides a user to configure a RAS, and performs automated configuration procedures on demand. Moreover, an HCoMS implements a “virtual wiring” concept that simplifies the interconnection of devices in software, with the effect that the RAS does not require any physical wiring to connect devices together. Furthermore, the HCoMS possesses advanced planning and scheduling capabilities to streamline production processes and aids in overall supply chain management. By implementing an HCoMS within a RAS will result in the SA manufacturing industry having reconfigurable systems that can handle the rapid introduction of new products, change-over between configurations with minimal effort, and achieving flexible manufacturing capacity without concern regarding fluctuating markets and market demands. These systems are more flexible in functionality; easier to configure; and are expertly monitored and controlled.

The long-term benefits of utilizing these novel systems include that production quantities and quality of products manufactured in SA will increase, while the price of manufactured products will decrease and the overall revenue increase. This will result in an increase of SA’s gross domestic product; restore the trust of foreign companies in SA; eventually draw investors and so increase the economic security in SA. In addition, these novel systems will require skilled personnel to develop, operate and maintain. This will create job opportunities for skilled professionals, which will further increase economic stability in SA.

Fundamentally, the implementation of RASs that utilizes an HCoMS will immensely benefit SA manufacturing companies by enabling them to successfully compete in global markets and meet the standards required by these markets. This finally raises the ultimate question. How can South African manufacturing industries not be more competitive, increase productivity, increase product variety, decrease lead times and ultimately show increased profits if they utilize a hybrid control and monitoring system developed for reconfigurable assembly systems?

References

[1] Y. Koren, "Reconfigurable Manufacturing Systems," *COMA annals,* vol. 1, pp. 69-79, 2004.

[2] N. F. Edmondson and A. H. Redford, "Generic Flexible Assembly System Design," *Assembly Automation,* vol. 22, pp. 139-152, 2002.

[3] W. Wang and Y. Koren, "Design Principles of Scalable Reconfigurable Manufacturing Systems," *IFAC Proceedings Volumes,* vol. 46, pp. 1411-1416, 2013.

[4] M. G. Mehrabi, A. G. Ulsoy, and Y. Koren, "Reconfigurable Manufacturing Systems: Key to Future Manufacturing," *Journal of Intelligent Manufacturing,* vol. 11, pp. 403-419, 2000.

[5] A. L. Andersen, C. Rösiö, J. Bruch, and M. Jackson, "Reconfigurable Manufacturing–An Enabler for a Production System Portfolio Approach," *Procedia CIRP,* vol. 52, pp. 139-144, 2016.

[6] K. K. Goyal, P. K. Jain, and M. Jain, "A Novel Methodology to Measure the Responsiveness of RMTs in Reconfigurable Manufacturing System," *Journal of Manufacturing Systems,* vol. 32, pp. 724-730, 2013.

[7] Y. Koren, "General RMS Characteristics. Comparison with Dedicated and Flexible Systems," in *Reconfigurable Manufacturing Systems and Transformable Factories*, ed Berlin, Heidelberg: Springer, 2006, pp. 27-45.

[8] Y. Koren, *The Global Manufacturing Revolution: Product-Process-Business Integration and Reconfigurable Systems* vol. 80: John Wiley & Sons, 2010.

[9] T. Lien and F. Rasch, "Hybrid Automatic-Manual Assembly Systems," *CIRP Annals-Manufacturing Technology,* vol. 50, pp. 21-24, 2001.

[10] J. Krüger, T. K. Lien, and A. Verl, "Cooperation of Human and Machines in Assembly Lines," *CIRP Annals - Manufacturing Technology,* vol. 58, pp. 628-646, 2009.

[11] D. Gyulai, Z. Vén, A. Pfeiffer, J. Váncza, and L. Monostori, "Matching Demand and System Structure in Reconfigurable Assembly Systems," *Procedia CIRP,* vol. 3, pp. 579-584, 2012.

[12] S. Takata and T. Hirano, "Human and Robot Allocation Method for Hybrid Assembly Systems," *CIRP Annals - Manufacturing Technology,* vol. 60, pp. 9-12, 2011.

[13] B. Lotter and H. P. Wiendahl, "Changeable and Reconfigurable Assembly Systems," in *Changeable and Reconfigurable Manufacturing Systems*, ed: Springer, 2009, pp. 127-142.

[14] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy*, et al.*, "Reconfigurable Manufacturing Systems," *CIRP Annals-Manufacturing Technology,* vol. 48, pp. 527-540, 1999.

[15] D. Minnich and F. Maier, *Supply Chain Responsiveness and Efficiency: Complementing or Contradicting Each Other?*: School of Business Administration, 2006.

[16] R.M. Setchi and N. Lagos, "Reconfigurability and Reconfigurable Manufacturing Systems - State of the Art Review," in *2nd IEEE International Conference of Industrial Informatics: Collaborative automation - One Key for Industrial Environments*, Berlin, 2004.

[17] H. P. Wiendahl, H. A. ElMaraghy, P. Nyhuis, M. F. Zäh, H. H. Wiendahl, N. Duffie*, et al.*, "Changeable Manufacturing-Classification, Design and Operation," *CIRP Annals-Manufacturing Technology,* vol. 56, pp. 783-809, 2007.

[18] M. N. Rooker, C. Sunder, A. Zoiltl, O. Hummer, and G. Ebenhofer, "Zero Downtime Reconfiguration of Distributed Automation Systems," in *Proceeding of the 3rd International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, Regensburg, 2007, pp. 326 - 337.

[19] H. ElMaraghy, "Flexible and Reconfigurable Manufacturing Systems Paradigms," *International Journal of Flexible Manufacturing Systems,* vol. 17, pp. 1-13, 2006.

[20] Y. Koren and M. Shpitalni, "Design of Reconfigurable Manufacturing Systems," *Journal of Manufacturing Systems,* vol. 29, pp. 130-141, 2010.

[21] R. Katz, "Design Principles of Reconfigurable Machines," *The International Journal of Advanced Manufacturing Technology,* vol. 34, pp. 430-439, 2007.

[22] M. Hentea, "Improving Security for SCADA Control Systems," *Interdisciplinary Journal of Information, Knowledge, and Management,* vol. 3, pp. 73-86, 2008.

[23] K. Stouffer, J. Falco, and K. Scarfone, "Guide to Industrial Control Systems (ICS) Security," *NIST Special Publication 800-82 Rev. 1,* 2013.

[24] A. Daneels and W. Salter, "What is SCADA?," in *International Conference on Accelerator and Large Experimental Physics Control Systems*, Trieste, Italy, 1999, pp. 339-343.

[25] R. M. van der Knijff, "Control Systems/SCADA Forensics, what's the Difference?," *Digital Investigation,* vol. 11, pp. 160-174, 2014.

[26] R. Kumar, "Recent Advances in SCADA Alarm System," *International Journal of Smart Home,* vol. 4, 2010.

[27] S. K. Shalini, B. Teshome, S. Muluneh, and B. Aragaw, "Working Phases of SCADA System for Power Distribution Networks," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering,* vol. 2, pp. 2037-2043, 2013.

[28] P. Zhang, *A Handbook for Engineers and Researchers: William Andrew*: William Andrew, 2008.

[29] W. T. Shaw, *Cybersecurity for SCADA Systems*: Pennwell Books, 2006.

[30] R. Radvanovsky and J. Brodsky, *Handbook of SCADA/Control Systems Security*, 2nd ed.: CRC Press, 2016.

[31] B. Hamed, "Implementation of Fully Automated Electricity for Large Building Using SCADA Tool like LabVIEW," *Current Trends in Technology and Sciences,* vol. 1, 2012.

[32] S. Karnouskos and A. W. Colombo, "Architecting the Next Generation of Service-Based SCADA/DCS System of Systems," in *IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society*, 2011, pp. 359-364.

[33] C. Alcaraz, R. Roman, P. Najera, and J. Lopez, "Security of Industrial Sensor Network-Based Remote Substations in the Context of the Internet of Things," *Ad Hoc Networks,* vol. 11, pp. 1091-1104, 2013.

[34] R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things," *IEEE Computer,* vol. 44, pp. 51-58, 2011.

[35] L. K. Ivert, "Advanced Planning and Scheduling Systems in Manufacturing Planning Processes," Department of Technology Management and Economics, Chalmers University of Technology, Sweden, 2009.

[36] H. Stadtler and C. Kilger, "Supply Chain Management and Advanced Planning - Concepts, Models, Software and Case Studies," 3rd ed: Springer, 2005.

[37] H. Stadtler, "Supply Chain Management and Advanced Planning—Basics, Overview and Challenges," *European Journal of Operational Research,* vol. 163, pp. 575-588, 2005.

[38] L. K. Ivert, *Use of Advanced Planning and Scheduling (APS) Systems to Support Manufacturing Planning and Control Processes*, ed.: Chalmers University of Technology, 2012.

[39] A. Tenhiälä and P. Helkiö, "Performance Effects of Using an ERP System for Manufacturing Planning and Control under Dynamic Market Requirements," *Journal of Operations Management,* vol. 36, pp. 147-164, 2015.

[40] K. Li, X. Zhang, J. Y.-T. Leung, and S.-L. Yang, "Parallel Machine Scheduling Problems in Green Manufacturing Industry," *Journal of Manufacturing Systems,* vol. 38, pp. 98-106, 2016.

[41] N. Pandhi, K. Singh, and S. Singh, "A Contrast: VSM, JIT, and MRP-II," *International Journal of Current Engineering and Technology,* vol. 5, pp. 385-396, 2015.

[42] A. Hossain and T. Zaman, "HMI design: An Analysis of a Good Display for Seamless Integration between User Understanding and Automatic Controls," in *2012 ASEE Annual Conference & Exposition*, 2012.

[43] Hexatec. (2010). *How to Design a Good HMI Display* [White Paper]. Available: http://www.hexatec.co.uk/Consultancy/hmi_display_design_guidelines.aspx

[44] D. Gersztenkorn and A. G. Lee, "Palinopsia Revamped: a Systematic Review of the Literature," *Survey of Ophthalmology,* vol. 60, pp. 1-35, 2015.

[45] S. Deodhar, P. Agrawal, and A. Helekar, "Effective Use of Colors in HMI Design," *Internasional Journal of Engineering Research and Applications,* vol. 4, pp. 384-387, 2014.

[46] D. Roessler and L. Garrison. (2013). *Five Practical Elements of Effective SCADA Graphics* [White paper]. Available: https://pgjonline.com/2013/02/19/five-practical-elements-of-effective-scada-graphics/

[47] OPTO 22. (2014). *Building an HMI that Works: New Best Practices for Operator Interface Design* [White paper]. Available: http://www.opto22.com/site/documents/doc_drilldown.aspx?aid=4351

[48] P. Gruhn, "Human Machine Interface (HMI) Design: The Good the Bad and the Ugly (and what makes them so)," *66th Annual Instrumentation Symposium for the Process Industries,* 27-29 Jan 2011.

[49] B. Hollifield, "High Performance HMI–Proof Testing in Real-World Trials," in *2013 ISA Water/Wastewater and Automatic Controls Symposium*, Orlando, Florida, USA, 2013.

[50] B. Hollifield, D. Oliver, I. Nimmo, and E. Habibi, *The High Performance HMI Handbook: A Comprehensive Guide to Designing, Implementing and Maintaining Effective HMIs for Industrial Plant Operations*, 1st ed., 2008.

[51] I. A. Mughal, L. Ali, N. Aziz, K. Mehmood, and N. Afzal, "Colour Vision Deficiency (CVD) in Medical Students," *Pak J Physiol,* vol. 9, pp. 9-1, 2013.

[52] B. Hollifield, "The High Performance HMI: Better Graphics for Operations Effectiveness," in *2012 Water/Wastewater and Automation Controls Symposium*, Orlando, Florida, USA, 2012.

[53] J. Krajewski, "Situational Awareness–The Next Leap in Industrial Human Machine Interface Design," *White paper, Invensys Systems, Houston, USA,* 2014.

[54] B. Pokharel, "Machine Vision and Object Sorting: PLC Communication with LabVIEW using OPC," Bachelor's Thesis, Department in Automation Engineering, HAMK University of Applied Sciences, 2013.

[55] J. Travis and J. Kring, "LabVIEW for Everyone: Graphical Programming Made Easy and Fun. 3rd," ed: New Jersey, Prentice Hall.

[56] R. H. Bishop, *LabVIEW 8 Student Edition*. Austin, Texas: Pearson Prentice Hall, 2007.

[57] H. M. Sabu, V. Aravind, A. Sullerey, and V. Binson, "Online Monitoring of PLC Based Pressure Control System," *International Journal of Research and Innovations in Science and Technology,* vol. 2, 2015.

[58] M. A. Muftah, A. M. Albagul, and A. M. Faraj, "Automatic Paint Mixing Process using LabVIEW," *Mathematics and Computers in Science and Industry,* pp. 233-238, 2014.

[59] J. Tomić, M. Kušljević, M. Vidaković, and V. Rajs, "Smart SCADA System for Urban Air Pollution Monitoring," *Measurement,* vol. 58, pp. 138-146, 2014.

[60] J. Niemann, "Development of a reconfigurable assembly system with enhanced control capabilities and virtual commissioning," Master Dissertation, Faculty of Engineering and Information Technology, Central University of Technology, Free State, 2013.

[61] C. Hayes, "Smartening up the Factory Floor [With Sensors]," *Engineering & Technology,* vol. 10, pp. 43-43, 2015.

[62] A. Gurhan, "Design and Development of Windows Store Application for Measurements and Monitoring," Master Thesis, Telemark University College, Faculty of Technology, Norway, 2013.

[63] V. Andreev, M. Malyutin, A. Karimov, and T. Karimov, "The Toolkit for Automation Design of Digital Systems with Parallel Architecture," in *Control and Communications (SIBCON), 2015 International Siberian Conference on*, 2015, pp. 1-3.

[64] Y. Shen and C. Lu, "Implementation of Fault Information Standardized Description and Network Transmission Based on LabVIEW," in *Proceedings of the Fourth International Conference on Information Science and Cloud Computing (ISCC2015). Guangzhou, China*, 2015.

[65] E. Vavilina and G. Gaigals, "Improved LabVIEW code generation," in *2015 IEEE 3rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, Riga, Latvia, 2015, pp. 1-4.

List of Publications

H. Vermaak and J. Niemann, “Validating a Reconfigurable Assembly System utilizing Virtual Commissioning,” in proceeding of the Pattern Recognition Association of South Africa and the Robotics and Mechatronics International Conference, 2015, Port Elizabeth. South Africa: ISBN: 978-1-4673-7449-1, pp 258 – 262.

H. Vermaak and J. Niemann, “Virtual Commissioning: A Tool to Ensure Effective System Integration,” to be presented in May 2017 at the 2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their application to Mechatronics (ECMSM) in San Sebastian, Spain.

Appendices

The following supporting video evidence can be acquired from the attached CD:

Appendix A: OPC Setup from CSV file

Appendix B1: Detection & Identification Module

Appendix B2: Order System Initial Test

Appendix B3: Order System Code Corrected

Appendix C: Test 2

Appendix D: Test 3