Development of an Automated Storage and Retrieval System in a Dynamic Knowledge Environment

---

Liam O' Shea BEng (Mech.)

MSc

Waterford Institute of Technology

Internal Supervisor: Mr. David Walsh

Submitted to Waterford Institute of Technology, June 2007.

Declaration

Development of an Automated Storage and Retrieval System in a Dynamic Knowledge Environment.

Presented to: **Mr. David Walsh**

Department of Engineering Technology

Waterford Institute of Technology

This Thesis is presented in fulfilment of the requirements for the degree of Masters of Science. It is entirely of my own work and has not been submitted to any other college or higher institution, or for any other academic award in this College. Where use has been made of the work of other people it has been fully acknowledged and fully referenced.

Signed: _______________

Liam O' Shea

Date: ______________

Abstract

This thesis summarises the development of an Automatic Storage and Retrieval System (ASRS) test-bed at Waterford Institute of Technology (WIT) within a dynamic knowledge environment.
The ASRS developed consists of control hardware and software communicating over a Fieldbus network. A simulation model of the WIT ASRS (capable of modelling any similar hi-bay storage system) and an order generator were also developed and these are linked to a database and a results spreadsheet.
This ASRS allows for a range of control strategies and order types to be investigated utilising the order generator and the database. There was also a facility developed which allows this mathematical model to run the actual requirements that the ASRS physical model works with, this allows for complete correlation between both models.
The development of a mathematical model plus a physical model ensures better understanding of ASRS making the sequence of operations obvious and helping to clarify the broad range of strategies to interested parties.
The best recorded performance was with current dwell point, simultaneous travel, dual control, free-nearest storage and nearest retrieval strategies selected in combination. In general, dual control improved performance (in terms of throughput), simultaneous travel was found to be better than rectilinear travel, dwell point at origin gave very poor results, and a dwell point at current, pick point or deposit point appears best.
Within the Knowledge environment a number of findings were made including: the development of a physical model is hugely beneficial to the acquisition of tacit knowledge by the chief researcher and supervisor and greatly benefits the development of undergraduate students (3 groups). The greater the volume of tacit and explicit knowledge available the faster the up-take for the students in a structured knowledge environment. There are great benefits to the college research centre by having a full time researcher in this area especially when the subject matter becomes complex, the learning curve for students can then be very steep.

Acknowledgements

I would like to thank those who gave help and advice during the course of this project. Their help proved invaluable.

I would like, in particular, to thank my supervisor, David Walsh for his continuous help, encouragement and guidance and also the following people whose help I greatly appreciated:

Mr. Joseph Phelan	Lecturer
Mr. Ned Cullinan	Lecturer
Mr. Paul Allen	Lecturer
Mr. Albert Byrne	Head of Department of Engineering Technology
Mr. Denis Moran	Head of School of Engineering
Mr. Mark Maher	Manufacturing Technician
Mr. Billy Walsh	Electronic Technician
Mr. Alan Nagle	Manufacturing Technician
Mr. Paul O' Mahoney	

I would especially like to thank my wife, Kay, for all her love and on-going life support and our 3 great daughters, Niamh, Roisin and Grainne for everything they have brought to my life. I would also like to acknowledge the help, encouragement and support I got from my parents, brothers and sisters.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction

The development of a dynamic integrated manufacturing environment has been on-going in WIT for the last 10 years with one major obstacle arising on a regular basis, that is, the retention of knowledge and experience gained by the individual researchers during the course of their research. This project is designed to utilise the knowledge-creating process to understand the dynamic nature of knowledge creation and to manage such a process effectively. This will allow the research staff to build more effectively on the previous research conducted on dynamic manufacturing in WIT and also to examine and explore the deep-rooted knowledge inherent in their system.

This lack of continuity in retaining knowledge assets is not altogether unique to research organisations; it is undoubtedly a major issue for many manufacturing bodies who would readily admit to an unhealthy reliance on their automation and process engineers with many stories circulating of major projects being seriously delayed and sometimes abandoned due to the movement of key personnel to other firms.

We are now living in a "knowledge-based society"; continuous innovation and the knowledge that enables such innovation have become important sources of sustainable competitive advantage. Knowledge and the ability to create and utilise knowledge is essential. However there is very little understanding of how organisations actually create and manage knowledge.

Specifically this project will concentrate on developing a test-bed for industrial Automated Storage and Retrieval System (ASRS) research, developing both a mathematical and a physical model based on an industrial ASRS in the locality. This facility will be capable of testing both mathematically and empirically a variety of ASRS control strategies including: dwell point, travel type, control, continuous or single operation, retrieval and storage strategies.

1.2 WIT Flexible Manufacturing Cell

Over the last ten years Waterford Institute of Technology (WIT) has developed a complex Flexible Manufacturing System (FMS) through a series of postgraduate projects (supported by a host of undergraduate projects). This FMS is currently producing two distinctive in-house products (Product 1 and Product 2).

Postgraduate research in the AMT Lab, began in 1995 when three projects [Crosse, 1997], [Maher, 1997] and [O' Connor, 1997] were set-up to continue with previous work carried out by undergraduate students. Since then numerous projects (approx. 300 student years of undergraduate projects and 20 years of full time postgraduate activity) have been undertaken with the goal of turning individual stand-alone industrial scale machines of various ages and configurations into a fully functional Flexible Manufacturing Cell (FMC) with the ambition one day of acquiring flexible manufacturing system status from the cell. The research concentrated primarily on the automated manufacturing of one product, Product 1, (Figure 1-1): The process involved the mechanical integration of the individual machines through the use of a conveyor system, and the development of SCADA and PLC control systems working with networked machine-PC interfaces. This single product production development is described in a series of postgraduate theses by [Crosse, 1997], [O' Connor, 1997], [Maher, 1997], [Mitchell, 1998], [McNelis, 2001] and a number of papers by [Phelan, 1997 & 2004].

Subsequent to this single product focus the emphasis switched to multiple products, with the introduction of Product 2, (Figure 1-2). "Engineering for Variety in a fully Automated Multi Production Manufacturing System" [Barry, pending]: describes the technical and operational development of the facility in the pursuit of this objective. Research conducted by Ross Alexander entitled "Engineering for Mixed Product Production in a Flexible Manufacturing and Flexible Assembly System" [Alexander, 2005] focused on development of the cell into a fully functional Mixed Product Flexible Manufacturing and Flexible Assembly System (FMAS). Running concurrently with this project were two associated postgraduate projects as follows: "Automated Storage and Retrieval: Upgrade of an industrial Hibay and Development of a College Based System", [O' Mahoney, 2004] and "Dynamic Scheduling, Artificial Intelligence and Control in a flexible Manufacturing System", [Flanagan, 2004].

The current status of the FMAS, after the conclusion of the three projects mentioned above, has it capable of producing two products with reasonable flexibility but without complete integration with a storage system. The dynamic scheduling module of the system is complete but requires a series of physical trials to be completed to ensure absolute confidence in its reasoning.

1.2.1 Product 1

This 2D product consists of nine Medium Density Fibre (MDF) blocks (64 mm x 64 mm x 12 mm). Eight of these have a slot machined in them to form the letter "M" (for manufacturing) when assembled, or when viewed the other way round they form the letter "W" (for WIT). The width of the slot on each is 9 mm. Each block has two 8.5 mm holes that are used to locate the block on a pallet (used for transportation between stations) by 2 corresponding pins, each of 8mm in diameter and held in place by a specific clamp. Figure 1-1 shows the composition for a completed Product 1, containing the nine parts.

1.2.2 Product 2

Product 2 is a cubic 3D structure made of 12mm softwood dowel. The structure consists of four uprights and 8 struts assembled and screwed together to form a three-dimensional cube. For production in the WIT system the components of Product 2 are separated onto 3 pallets and held together in a jig and clamp arrangement specifically designed for the purpose. The four uprights are machined together on one pallet and the eight struts are separated equally onto the two remaining pallets. Figure 1-2 shows a complete Product 2.

1.2.3 Flexible Manufacturing System

The term "Flexible Manufacturing *Cell*" is commonly used to refer to a machine grouping that consists of either manually operated or automated machines, or a combination of the two. The cell may or may not include automated material handling, and it may or may not be computer controlled. The term "flexible manufacturing *system*" generally means a fully automated system consisting of automated workstations, automated materials handling, and computer control [Groover, 2001]. In contrast to this Rembold et al (1993) defines a FMC as an automated computer controlled cell, and a FMC with the addition of Automated Storage and Retrieval as a FMS.

The three basic elements of any FMS: workstations, material transport and storage system, computer control system, are detailed below, with a breakdown in the context of the WIT system.

The FMC workstations in WIT (see Figure 1-3) are as follows:

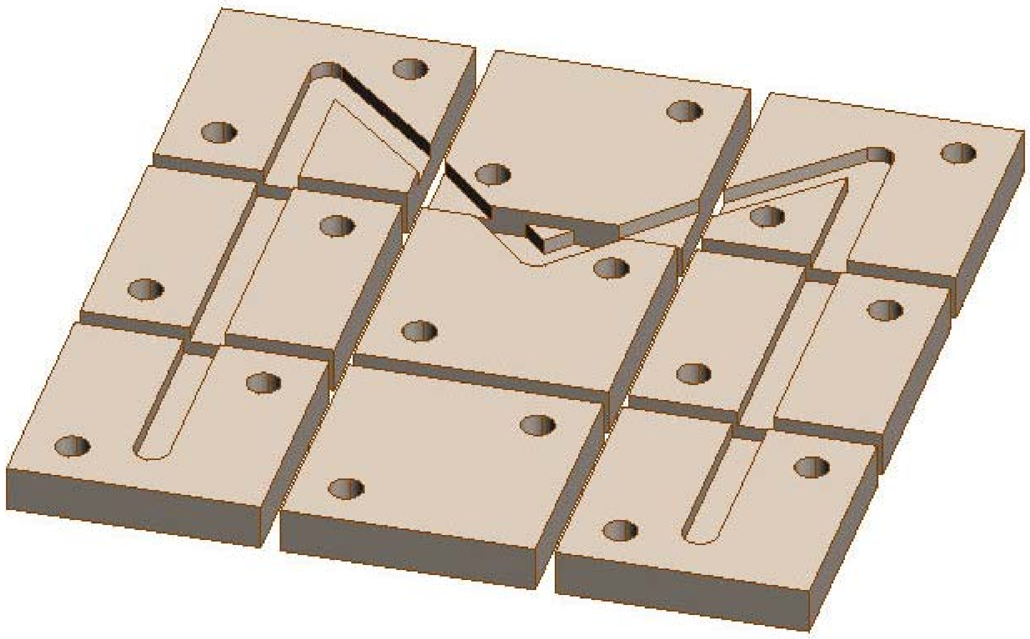


Figure 1-1: Product 1

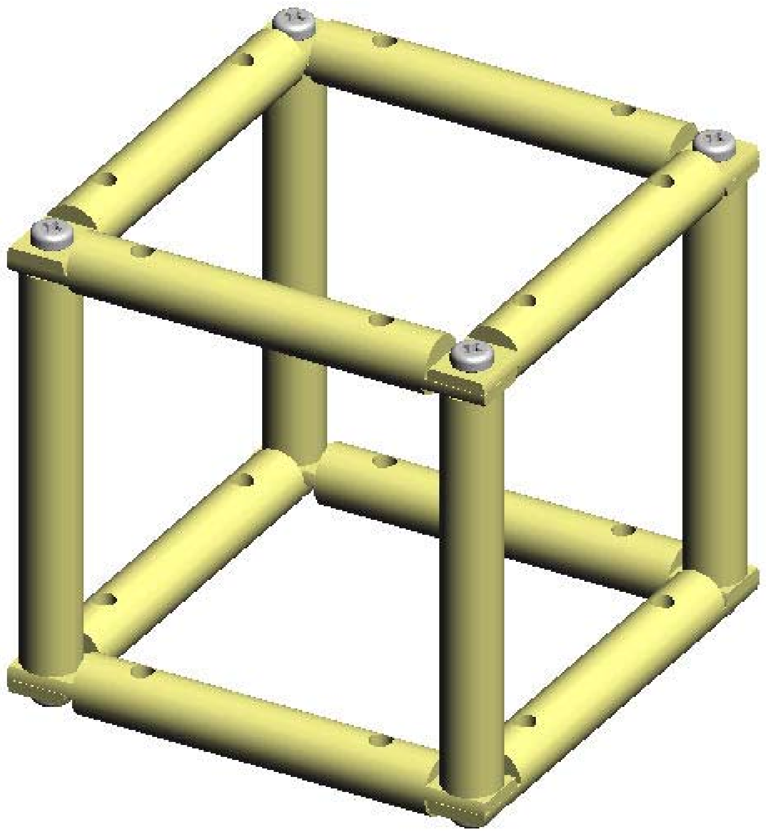


Figure 1-2: Product 2

- Holke Milling Machine: A 3-axis CNC milling machine with a retrofitted controller, networked to the cell controller via an in-house built interface unit attached to the Holke PC. Assigned to the production of Product 1.
- Deckel Milling Machine: A 3-axis CNC milling machine, networked to the cell controller via an in-house built interface unit attached to the Deckel PC. Assigned to the production of Products 1 and 2.
- CMM: A Kemco coordinate measuring machine (originally manual: some degree of automation achieved during a previous undergraduate project), networked to the cell controller via the CMM PC through an in-house developed I/O card and LabView software, set-up for the measurement of Product 1 only.
- Load/Unload Station: This station consists of a Bosch SCARA pick and place robot which feeds product into the FMS and removes completed or unwanted product.
- Assembly Station: This workstation consists of a Staubli robot with an automatic screw feeder and driver attached, and a complex dedicated pneumatic Assembly Station for the assembly of Product 2.

The FMC materials transport and storage system consists of:

- A precision palletised conveyor system (Bosch).
- A materials stacking system for storing blanks for Product 1 (raw material stacker).
- A materials stacking system for storing completed Product 1(finished goods stacker).
- Bosch load/ unload Robot, (SCARA construction).

The FMC computer control system consists of:

- A Mitsubishi PLC
- A Cell controller (referred to as "cellhost", running Wonderware Intouch SCADA)
- A Windows NT server linking individual workstations to the cell controller
- A Windows NT workstation which holds the SQL database.

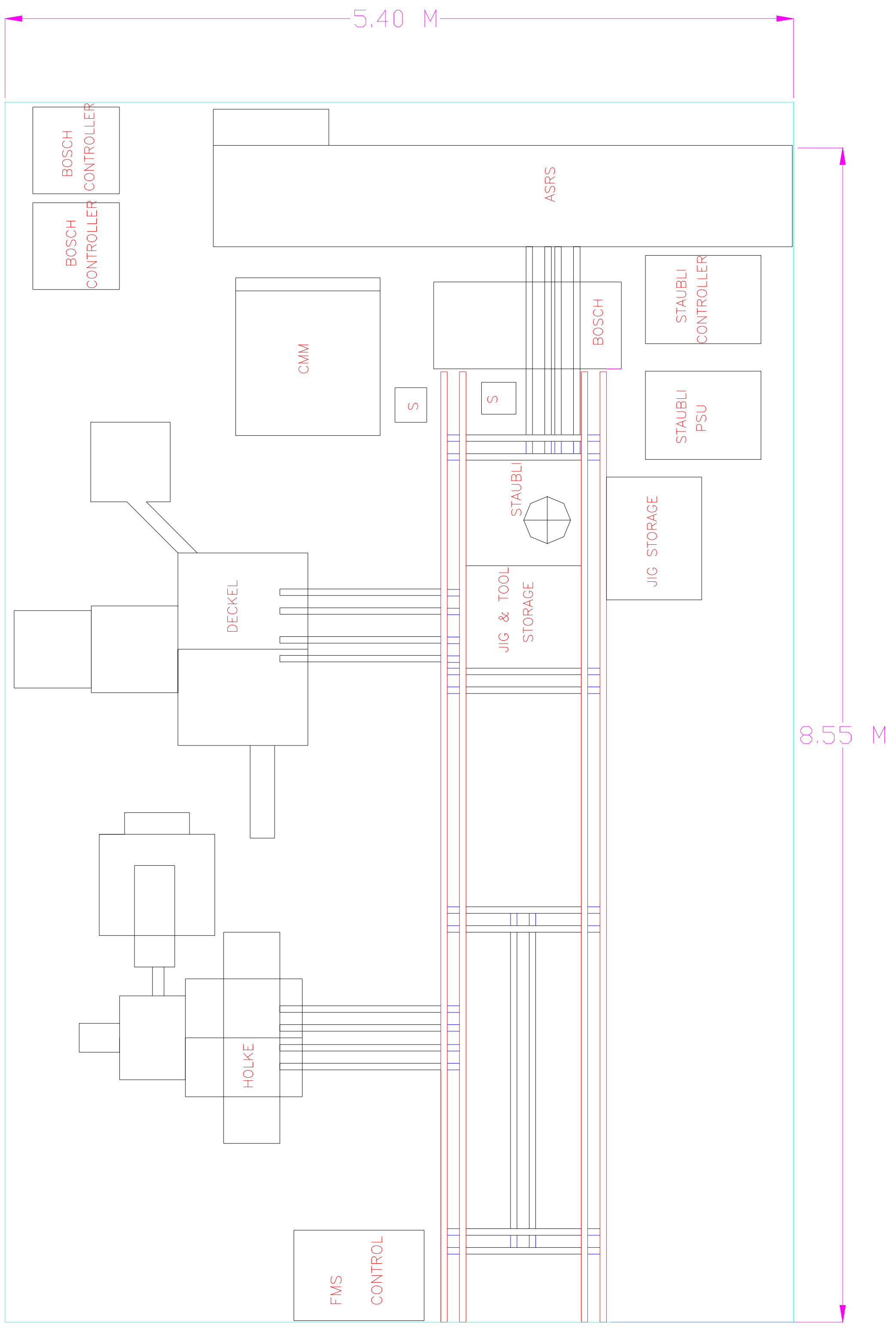


Figure 1-3: Physical layout of the cell

1.2.4 ASRS Development

The long-term vision for the FMS included a fully automatic storage and retrieval system that would be completely integrated into the FMAS. With this in mind the first ASRS in WIT was built during a series of undergraduate projects beginning in 1998. This ASRS was constructed of extruded aluminium conveyor section that was bolted together using standard bracketing and connected to a supporting brick wall. This set-up proved to be unstable. Two pneumatic cylinders were connected in series to perform pick and place operations. This unit was attached to the belt of a vertically mounted conveyor to enable vertical movement. The assembly was moved horizontally through attachment to two horizontally mounted conveyors. BOSCH 3-phase motors powered the horizontal and vertical movements. These motors were designed for use with long conveyor sections and were oversize for this application. Basic control was achieved through a GE Fanuc 90/30 PLC and some basic electrical switchgear.

Contacts were made with a local company who, at that stage (2002), were considering the upgrade of their own Hi-Bay storage system and agreement was reached that a joint approach would be taken to the up-grade of both storage and retrieval systems. It is for this reason that great emphasis has been placed on the correlation of both systems in terms of hardware chosen and in the control system developed.

The initial work on this new WIT ASRS was conducted as part of a postgraduate research project [O' Mahoney, 2004] with the current author acting as supervisor to this work. Great progress was made during the course of this research project (quite a lot of the physical build was completed), which is outlined in Chapter 3 and it is the aim of this current project to further develop the WIT ASRS to ensure that it fulfils its function as a test-bed for control strategy development.

1.3 Aims and Objectives of Project

The objective of this project was to develop a test-bed for ongoing research in ASRS design allowing for the analysis of a variety of control strategies both mathematically and empirically within a knowledge paradigm of knowledge acquisition, tacit knowledge development, tacit knowledge extension, explicit knowledge extension and knowledge spiralling. The following aims were identified in order to achieve this:

- Develop an understanding of industrial storage and retrieval systems.

- Develop fieldbus based control utilising SCADA software and integrate this hardware with a control database and an order generator.

- Develop an entirely flexible ASRS control system which would allow for the empirical testing of a wide range of control strategies incorporating a mathematical model to generate a broad spectrum of requirements in terms of parts, operation and time required.

- Utilise a mathematical model of the ASRS in combination with output to a spreadsheet to confirm the findings from the physical model and to produce long-term data on performance etc.

- Carry out extensive testing, troubleshooting and trials on both models reviewing the results and making recommendations on most suitable settings for a variety of requirements on the ASRS.

- Develop a deep understanding of the existing ASRS and capture and elaborate on this as a model in knowledge creation.

- Showcase to the manufacturing and research sectors the benefits of the knowledge creation process as a practical solution to the on-going problem of poor retention of knowledge assets.

1.4 Summary

This thesis summarises the development of a highly flexible ASRS test-rig which allows for the testing of a variety of control strategies for an unlimited range of demands. It details the application of a SIMUL8^{TM} mathematical model of the ASRS to confirm the results obtained from the physical model. It makes recommendations on strategy, in a limited sense, to be employed to best suit the demand. This work has been conducted as a physical demonstration of a knowledge creation paradigm and attempts to demonstrate the practical benefits of knowledge creation in a dynamic knowledge environment.

2 ASRS Development in a Knowledge Environment

2.1 Introduction

This chapter summarises the current status of ASRS development highlighting the most recent research in this area and then goes on to discuss the topical area of knowledge management and creation.

Automated storage and retrieval systems were first introduced in the 1950s to eliminate the walking that accounted for 70 % of manual retrieval time [Groover, 2001]. There have been many advancements in ASRS technology in the last forty years and their number is predicted to grow rapidly in the next decade.

ASRS have many benefits including savings in labour costs, improved material flow and inventory control, improved throughput level, high floor-space utilisation, increased safety and stock rotation.

The efficient operation of ASRS requires planning of (a) physical storage specifications: height, length, width of storage structure and storage opening, (b) operating characteristics of AS/R systems: horizontal and vertical velocity, acceleration rate and number of machines and (c) control strategy.

Typically, ASRS consist of a series of storage aisles each of which is served by a storage and retrieval (S/R) machine or crane. Each aisle is supported by a pickup and delivery (P&D) station typically located at the end of the aisle and accessed by the S/R machine and the external handling system.

Applications of ASRS exist in the assemblies of small electronic components where assembly work-stations are installed in the openings of the storage racks, in clean-room manufacturing environments to reduce the contamination of the products from manual handling, in healthcare distribution centres where pallet loads of medical products, ranging from IV solutions to heart valves are temporarily stored for later distribution. Frozen food processing environments where temperature is always kept at -29°C, making it extremely hostile to human operators represent other implementations of the ASRS.

A recent application of the ASRS is in the automotive industry. After car bodies have been painted, they are moved into storage in an ASRS to coordinate the production schedule with the number of bodies painted a specific colour. The selected bodies are then retrieved and returned to production.

When it comes to the subject of knowledge creation countries are now marketing themselves as having a knowledge-based economy. What exactly does this mean? There are many definitions, but an OECD report "The Knowledge-Based Economy" (1996) defines them as economies which are directly based on the production, distribution and use of knowledge and information, leading towards growth in high-technology investments, high-technology industries, more highly skilled labour and associated productivity gains.

This chapter attempts to define knowledge, its value and its relationship with technology. It also concentrates on the various dimensions of knowledge and the development of knowledge through learning.

2.2 Objectives for automating a Company's Storage Operations

A list of possible objectives that a company may want to achieve by automating its storage operations is shown below [Groover, 2001]:

- To increase storage capacity
- To increase storage density
- To recover factory floor space presently used for storing work-in-process.
- To improve security and reduce pilferage
- To reduce labour cost and/or increase labour productivity in storage operations
- To improve safety in the storage function
- To improve control over inventories
- To improve stock rotation
- To improve customer service
- To increase throughput

2.3 Components and Operating Features of an ASRS

The content of this section is in general attributable to Groover (2001).

Virtually all ASRS consist of the following components:

1. Storage structure
2. S/R machine
3. Storage modules
4. P&D stations
5. Control system

2.3.1 Storage Structure

The storage structure is the rack framework typically made of fabricated steel, which supports the loads contained in the ASRS. This structure must possess sufficient strength and rigidity that it does not deflect significantly due to the loads in storage or other forces on the framework. The individual storage compartments in the structure must be designed to accept and hold the storage modules used to contain the stored materials. The rack structure may also be used to support the roof and siding of the building in which the ASRS resides. Another function of the storage structure is to support the aisle hardware required to align the S/R machines with respect to the storage compartments of the ASRS. This hardware includes guide rails at the top and bottom of the structure as well as end stops and other features required to provide safe operation.

2.3.2 S/R Machine

The S/R machine is used to accomplish storage transactions, delivering loads from the input station into storage and retrieving loads from storage and delivering them to the output station. To perform these transactions the S/R machine must be capable of horizontal and vertical travel to align its carriage (which carries the load) with the storage compartment in the rack structure. In many cases the S/R machine consists of a rigid mast on which is mounted a rail system for vertical motion of the carriage. Wheels are attached at the base of the mast to permit horizontal travel along a rail system that runs the length of the aisle. A parallel rail at the top of the storage structure is used to maintain alignment of the mast and carriage with respect to the rack structure.

The carriage includes a shuttle mechanism to move loads into and from their storage compartments. The design of the shuttle system must also permit loads to be transferred from the S/R machine to the Pick and Deposit (P&D) station or other material handling interface with the ASRS. The carriage and shuttle are positioned and actuated automatically in the usual ASRS. Man-on-board S/R machines are equipped for a human operator to ride on the carriage.

To accomplish the desired motions of the S/R machine, three drive systems are required: horizontal movement of the mast, vertical movement of the carriage and shuttle transfer between the carriage and a storage compartment. Modern S/R machines are available with horizontal speeds up to 200 m/min along the aisle and vertical or lift speeds up to around 50 m/min. These speeds determine the time required for the carriage to travel from the P&D station to a particular location in the storage aisle. Acceleration and deceleration have a more significant impact on travel time over short distances. The shuttle transfer is accomplished by any of several mechanisms, including forks (for pallet loads) and friction devices for flat bottom tote bins.

2.3.3 Storage Modules

The storage modules are the unit load containers of the stored material. These include pallets, steel wire baskets and containers, plastic tote bins and special drawers (used in mini-load systems). These modules are generally made to a standard base size that can be handled automatically by the carriage shuttle of the S/R machine. The standard size is also designed to fit in the storage compartments of the rack structure.

2.3.4 Pick and Deposit Stations

The pick and deposit station is where loads are transferred into and out of the ASRS. They are generally located at the end of the aisles for access by the external handling system that brings loads to the ASRS and takes loads away. Pickup stations and deposit stations may be located at opposite ends of the storage aisle or combined at the same location. This depends on the origination point of incoming loads and the destination of output loads. A P&D station must be designed to be compatible with both the S/R machine shuttle and the external handling system. Common methods to handle loads at the P&D station include manual load / unload, forklift truck, conveyor (e.g. roller) and AGVs.

2.3.5 Control System

The principle ASRS control problem is positioning the S/R machine within an acceptable tolerance at a storage compartment in the rack structure to deposit or retrieve a load. The locations of materials stored in the system must be determined to direct the S/R machine to a particular storage compartment. Within a given aisle in the ASRS each compartment is identified by its horizontal and vertical positions and whether it is on the right side or left side of the aisle. A scheme based on alpha-numeric codes can be used for this purpose. Using this location identification scheme, each unit of material stored in the system can be referenced to a particular location in the aisle. The record of these locations is called the ‘item location file’. Each time a storage transaction is completed the transaction must be recorded into the item location file.

Given a specified storage compartment to go to, the S/R machine must be controlled to move to that location and position the shuttle for load transfer. One positioning method uses a counting procedure in which the number of bays and levels are counted in the direction of travel (horizontally and vertically) to determine position. An alternative method is a numerical identification procedure in which each compartment is provided with a reflective target with binary-coded location identifications on its face. Optical scanners are used to read the target and position the shuttle for depositing or retrieving a load.

Computer controls and programmable logic controllers are used to determine the required location and guide the S/R machine to its destination. Computer control permits the physical operation of the ASRS to be integrated with the supporting information and record-keeping system. Storage transactions can be entered in real-time, inventory records can be accurately maintained, system performance can be monitored and communications can be facilitated with other factory computer systems. These automatic controls can be superseded or supplemented by manual controls when required under emergency conditions or for man-on-board operation of the machine.

2.4 Operation of an ASRS

An ASRS machine usually operates in one of two modes: single cycle (SC) or dual cycle (DC) also known as Interleaving. For each of the modes the S/R machine starts at the P&D station, stores and/or retrieves a load, and returns to the P&D station to complete a cycle. In a SC the S/R machine either stores or retrieves, while in a DC it both stores and retrieves in one cycle. In a DC, the S/R machine picks up a load from a P/D station, travels to a storage location to store it, travels to another location to retrieve a load and then returns to the P&D station to deliver it.

According to Han et al (1987) the effectiveness of an ASRS depends on the methods of control that govern the scheduling of storages and retrievals. A common practice in sequencing storage and retrieval requests is that both requests are processed in a first-come-first-served (FCFS) manner. The FCFS assumption is reasonable for storages, since most ASRS are interfaced with a conveyor loop for input and output. In this case, it is difficult to change the sequence of loads presented for storage. However, the

FCFS assumption is less compelling for retrievals since retrieval requests are just electronic messages and can be easily re-sequenced.
In a DC, storage and retrieval requests can be paired to decrease the time spent travelling between the storage and retrieval locations. By minimising the travel time, it is possible to increase system throughput (i.e. the number of storages or retrievals performed per period) and reduce ASRS operating costs such as wear of mechanical parts and electric power cost. Han et al (1987) claim that a 50% or more decrease in the travel-between time component of a dual cycle leads to an increase in throughput of 10-15%. Such an increase in throughput could help to handle peak demand in the operation phase and eliminate an aisle in a multi-aisle system in the design phase, which would lead to considerable savings.

2.5 ASRS Storage Policies

In an ASRS empty storage locations are assigned to an incoming pallet in different ways. In random storage assignment, a pallet has an equal chance of being stored in any of the open locations. In a class-based storage assignment the products and storage racks are divided into a number of classes according to the product turnover frequencies. The highest turnover product is stored in the class of storage rack closest to the input / output point (P&D location). A pallet is stored randomly within the class. In dedicated storage each product is assigned to a specific location or set of locations in the storage rack again according to their turnover frequencies.
White and Kinney (1982) noted that in comparison to dedicated storage, random storage generally requires less storage space because the maximum aggregate storage requirement is generally less than the aggregate maximum storage requirements for each product in storage. In comparison to random storage, dedicated storage results in reduced travel time if equal storage areas are assumed. However, since the class-based and dedicated storage policies are based on turnover frequency for each product it is difficult to use them if the turnover frequencies of the products vary with time. Random storage policy is not affected by varying turnover frequencies.

2.5.1 Storage Assignment and Interleaving Rules

Storage assignment is the selection of an open rack location for the storage of an arriving pallet [Graves et al, 1977]. Interleaving or Dual Cycle operation allows for the completion of both a store request and a retrieve request on a single trip from the

P&D point. That is, upon completion of a store the S/R machine will not return empty to the P&D point for its next instruction; instead the crane will move (interleave) to the location of a retrieve request, make the retrieval and then return to the P&D point. Interleaving systems are also known as dual-address systems, since the S/R machine is capable of visiting two locations (or addresses) between successive returns to the P&D point.

2.5.1.1 Storage Assignment Rules

1. Random storage assignment (RAN): The storage location is chosen randomly from all open rack locations. This rule has been used to approximate the performance of the closest-open-location (COL) rule, a rule widely used in practice.
2. Class-based storage assignment (C2 or C3): The items and the rack locations are ranked according to turnover and distance (in travel time) from the P&D point, respectively. These ranked lists are then partitioned into a small number of matched classes (2 or 3) such that the class of items with the highest turnover is assigned randomly within the class of locations closest to the P&D point, etc.
3. Full turnover-based storage assignment (Full): For this rule the highest turnover item is assigned to the location closest to the P&D point. This rule represents the limit of class-based rules.

2.5.1.2 Interleaving Rules

1. No interleaving (NIL): All storage and retrieval requests are initiated with the S/R at the P&D point. These are sometimes referred to as ‘single address’ or single cycle systems because the S/R unit is only capable of visiting a single rack location (address) between successive returns to the P&D point.
2. Mandatory interleaving with FCFS queue discipline of retrieves (MIL/FCFS): A retrieve is performed every time a store is made and the retrieve is chosen FCFS from the retrieve queue.
3. Mandatory interleaving with selection queue of K retrieves (MIL/Q=K): This rule is applicable only when a class-based storage assignment rule is used. Again, a retrieve is performed every time a store is made; however, the retrieve is selected from the first K entries in the retrieve queue. These

K retrieves are searched until a retrieve of the same class as the previous store is found. If a retrieve from the same class is not found, the search is repeated using the ‘next best’ class.

2.6 ASRS Performance

The performance of ASRS varies by the definition of the measure and the operating policies adapted [Elsayed & Lee, 1996]. Measures of performance may include:

1. The travel time per storage/retrieval request
2. The total time required to store/retrieve a batch of orders
3. The average waiting time for a storage/retrieval request

Many parameters affect the performance of the ASRS. Although some of the parameters are interrelated, they are divided into three groups: demand requirements, physical design and operating policies.

Demand requirements represent the orders that need to be stored or retrieved to meet the required production (distribution) schedule. The demand may be defined by several parameters:

(i) Number of orders received per unit time.

(ii) The pattern of retrieving the demand as it arrives to the ASRS: A static retrieval pattern implies that when demand arrives it is accumulated into one group and then the storage and retrieval processes are performed on the group until all orders are completed. New arrivals, while a group of storage and retrieval is being processed, form a different group that can be processed after the completion of the current group. A dynamic retrieval pattern implies that a new arrival during the processing of a group is added to the group and re-sequencing and batching of orders is made to accommodate the new arrival(s).

(iii) Number of items to be stored or retrieved per order.

(iv) Weights and sizes of items to be processed.

(v) The due date of the orders.

The second group of parameters that affect the performance of ASRS relate to its physical design. Some of these parameters are: size of storage bins, length and height of storage structure (building the aisle too long may cause the S/R machines to operate at too high a percentage of their capacity), single or double deep rack, and capacity and number of S/R machines.

The third group of parameters that affect the performance of ASRS are the operating policies of the system, which involve rules for storage and retrieval (storage cycle, retrieval cycle, storage and retrieval in the same cycle) of materials, turnover time and item popularity, order sequencing and batching, order retrieval policies (FCFS, LCFS, priority, etc.), order storage policies and routing of the S/R machine.

2.7 Dwell Point Analysis

The method of determining the point to position the S/R machines when idle is referred to as dwell point policy and the point where the S/R machine is positioned as the dwell point [Egbelu & Wu, 1993].

They also state that in positioning the S/R machine when idle a properly selected dwell point policy will reduce travel time of the S/R machine in warehouse operation. Several dwell point policies are available. These dwell point rules are derived from simple rules-of-thumb or mathematical programming. Some of these rules are static in nature while others respond dynamically to changes in storage and retrieval demand. Typical dwell point rules include:

(1) Dynamically position the S/R machine at a location that minimises the expected S/R machine travel or response time from the dwell point to the points of need.

(2) Dynamically position the S/R machine at a location that minimises the maximum S/R machine travel or response time from the dwell point to the points of need.

(3) Always position the S/R machine at the input station whenever idle.

(4) Always position the S/R machine at the output station whenever idle.

(5) Always position the S/R machine at the mid-point location in the rack whenever idle.

(6) Dynamically position the S/R machine at the last location it visited following the completion of either a single command or dual command cycle.

The dynamic dwell point rules (1 and 2) were proposed by Egbelu (1991). These two rules recognise the dynamic fluctuation in the storage and retrieval demands that are experienced in ASRS from one scheduling period to another. A period may represent an hour, a shift or a day depending on the production schedule of the shop or the distribution centre served by the ASRS. A linear programming model based on location theory was presented by Egbulu (1991) to minimise the service response time in an ASRS through the optimal selection of the dwell point of the S/R machine when idle. For dwell point rule (1) the objective is to minimise the expected travel time or response time of the S/R machine to the location where it is needed, given that the machine originates from the dwell point. For dwell point rule (2) the objective is the minimisation of the maximum travel time to the point of need, again assuming that the machine originates from the dwell point.

The dwell point rules (3 – 5) are static in nature and are therefore time, traffic and situation invariant. These rules are mainly concerned with selecting a point along the aisle where the S/R machine should be positioned. In this respect, these rules consider the problem as a one-dimensional location problem. Practically, in an ASRS system, the decision is not only to determine the point along the horizontal guide track to dwell the machine, but also to specify how high the retrieval arm should be positioned. The position of the retrieval arm is important since the time required for the S/R machine to reach a point is determined by the longer of either the horizontal travel time or the vertical travel time. In rules (1) and (2) the S/R dwell point selection problem is viewed as a two-dimensional location problem in which the position of the machine on the linear track and the position of the arm must both be determined simultaneously.

As would be expected, traffic intensity influences the proportion of the time the S/R machine remains idle, and consequently, the frequency with which the dwell point algorithm is invoked. The lower the traffic rate, the higher the frequency of invoking the dwell point algorithm.

The dwell point rule (6), positioning of the S/R machine at the last location visited, does not really respond to the dynamic changes in storage and retrieval demands brought about by the changing production schedule. Rather, it is a function of the sequencing of the storage and retrieval requests made to the ASRS.

Traditionally, the dwell point selection uses simple rules-of-thumb (rules (3-6) previously described). These four rules are static in nature as they do not consider the fluctuation in the level of activities in the ASRS from period to period. Egbelu (1991) proposed two dwell point rules that are dynamic in nature. These two rules use linear programming models to dynamically determine the dwell point.

2.8 Sizing the ASRS Rack Structure

The total storage capacity of one storage aisle depends on how many storage compartments are arranged horizontally and vertically in the aisle [Groover, 2001]. This can be expressed as follows:

Capacity per aisle = $2n_yn_z$ **Equation 2-1**

Where:

n_y = number of load compartments along the length of the aisle

n_z = number of load compartments that make up the height of the aisle

The constant, 2, accounts for the fact that loads are contained on both sides of the aisle.

If a standard size compartment is assumed (to accept a standard size unit load), then the compartment dimensions facing the aisle must be larger than the unit load dimensions. Let x and y = the depth and width dimensions of a unit load (e.g. a standard pallet size) and z = the height of the unit load. The width, length and height of the rack structure of the ASRS aisle are related to the unit load dimensions and number of compartments as follows:

$W = 3(x + a)$ **Equation 2-2**

$L = n_y(y + b)$ **Equation 2-3**

$H = n_z(z + c)$ **Equation 2-4**

Where:

W, L and H = width, length and height of one aisle of the ASRS rack structure respectively

x, y and z = the dimensions of the unit load

a, b and c = allowances designed into each storage compartment to provide clearance for the unit load and to account for the size of the supporting beams in the rack structure

For the case of unit loads contained on standard pallets, Groover (2001) recommends values for the allowances as: a = 150mm, b = 200mm and c = 250mm. For an ASRS with multiple aisles, W is simply multiplied by the number of aisles to obtain the overall width of the storage system. The rack structure is built above floor level by 300 – 600 mm and the length of the ASRS extends beyond the rack structure to provide space for the P&D station.

2.9 ASRS Throughput

System throughput is defined as the hourly rate of S/R transactions that the automated storage system can perform [Groover, 2001]. A transaction involves depositing a load into storage or retrieving a load from storage. Either one of these transactions alone is accomplished in a single command cycle. A dual command cycle accomplishes both transaction types in one cycle: since this reduces travel time per transaction, throughput is increased by using dual command cycles when the dwell point is specified as other than 'Current' or 'Deposit Point'.

Several methods are available to compute ASRS cycle times to estimate throughput performance. The method presented here is recommended by the Materials Handling Institute as summarised by Groover (2001). It assumes:

a. Randomised storage of loads in the ASRS (i.e. any compartment in the storage aisle is equally likely to be selected for a transaction)

b. Storage compartments are of equal size

c. The P&D station is located at the base and end of the aisle

d. Constant horizontal and vertical speeds of the S/R machine

e. Simultaneous horizontal and vertical travel

For a single command cycle, the load to be entered or retrieved is assumed to be located at the center of the rack structure. Thus, the S/R machine must travel half the length and half the height of the ASRS, and it must return the same distance. The single command cycle time can therefore be expressed by:

$$T_{cs} = 2 * Max\left\{\frac{0.5L}{v_y}, \frac{0.5H}{v_z}\right\} + 2T_{pd} = Max\left\{\frac{L}{v_y}, \frac{H}{v_z}\right\} + 2T_{pd} \qquad \textbf{Equation 2-5}$$

Where:

T_{cs} = cycle time of a single command cycle (min/cycle)

L = length of the ASRS rack structure (m)

v_y = velocity of the S/R machine along the length of the ASRS (m/min)

H = height of the rack structure (m)

v_z = velocity of the S/R machine in the vertical direction of the ASRS (m/min)

T_{pd} = pickup and deposit time

Two P&D times are required per cycle, representing load transfers to and from the S/R machine.

For a dual command cycle, the S/R machine is assumed to travel to the center of the rack structure to deposit a load and then it travels to ¾ the length and height of the

ASRS to retrieve a load. Thus the total distance travelled by the S/R machine is ¾ the length and ¾ the height of the rack structure and back. In this case cycle time is given by:

$$T_{cd} = 2 * Max\left\{\frac{0.75L}{v_y}, \frac{0.75H}{v_z}\right\} + 4T_{pd} = Max\left\{\frac{1.5L}{v_y}, \frac{1.5H}{v_z}\right\} + 4T_{pd}$$

Equation 2-6

Where:

T_{cd} = cycle time for a dual command cycle (min/cycle)

System throughput depends on the relative numbers of single and dual command cycles performed by the system. Let R_{cs} = number of single command cycles performed per hour and R_{cd} = number of dual command cycles per hour at a specified or assumed utilisation level. The equation for the amount of time spent in performing single and dual command cycles each hour is:

$$R_{cs}T_{cs} + R_{cd}T_{cd} = 60U$$ **Equation 2-7**

Where:

U = system utilisation during the hour

The right hand side of the equation gives the total number of minutes of operation per hour. To solve this equation the relative proportions of R_{cs} and R_{cd} must be determined, or assumptions about these proportions must be made. Then the total hourly rate is given by:

$$R_c = R_{cs} + R_{cd}$$ **Equation 2-8**

Where:

R_c = total S/R cycle rate (cycles/hr)

Note that the total number of storage and retrieval transactions per hour will be greater than this value unless R_{cd} = 0, since there are two transactions accomplished in each dual command cycle. Let R_t = the total number of transactions performed per hour; then:

$$R_t = R_{cs} + 2R_{cd}$$ **Equation 2-9**

2.10 Current ASRS Research

Lee et al (1997) studied the effect of sequencing storage and/or retrieval requests on the reduction of travel time by a S/R machine, and consequently on throughput, for ASRS where storage locations are predetermined. They found that different types of

ASRS (random, dedicated) require different sequencing to be employed and that the proposed sequencing methods can significantly affect performance, with dynamic assignment methods clearly outperforming the static methods achieving about 10-20% reduction of expected travel-between time for dual cycles.

Mansuri et al (1997) present an algorithm which provides a means to investigate a variety of dedicated type of storage allocation alternatives. A computer program, developed in QuickBasic, facilitates the ASRS planning process and also allows ‘what if’ type of questions to be evaluated through alternative plans. This program can also be used as a tool to determine the initial configuration and specifications of a storage structure and operating policy.

Keserla & Peters (1994) have analysed a dual shuttle automated storage and retrieval system with the following results:

- Throughput improvements in the range of 40-45% obtained using a quadruple command cycle relative to dual command cycles with a single shuttle system.
- With the dual shuttle design, travel between is virtually eliminated for a dual command cycle.

This dual shuttle system seems to show promise for situations requiring high throughput. The main disadvantage with the new design is the cost of the extra S/R machine.

Moon and Kim (2001) report on research to find the tolerance limits and appropriate strategies for random and class-based storage policies of ASRS as applied to production plan changes. They conclude that random storage policy performs well with production quantity variations. This policy tolerates up to 60 % of the variations.

Han et al (1987) address the issue of throughput improvement by retrieval sequencing in conventional unit load automated storage/retrieval systems when several retrieval requests are available and dual command cycles are performed. First-come-first-served is taken as the reference sequencing rule. The following results are presented: 10-15% improvement in throughput can be obtained by reducing the travel-between component of dual command cycle by 50% or more.

An equation is derived for approximating the mean dual command cycle time using a nearest-neighbour sequencing heuristic. A further equation is presented for a lower bound on the mean dual command cycle time for any block sequencing rule. The nearest-neighbour heuristic obtains average throughput within 5-8% of the maximum possible average throughput.

Hausman et al (1976) deal with optimal storage assignment. Results are obtained which compare the operating performance of three storage assignment rules: random assignment, which is similar to the closest-open-location rule used by many currently operating systems; full turnover-based assignment; and class-based turnover assignment. It is shown that significant reductions in crane travel time (and distance) are obtainable from class-based turnover-based rules rather than closest-open-location (essentially random) policies. These improvements can, under certain circumstances, be directly translated into increased throughput capacity for existing systems and may be used to alter the design (e.g. size and number of racks, speed of cranes, etc.) of proposed systems in order to achieve a more desirable system balance between throughput and storage capacity.

Muralidharan et al (1995) highlight the development of a new shuffling heuristic-based approach that combines random storage and class-based storage. In shuffling, random storage is employed for the storage location assignment, but, when the S/R machine is idle, shuffling shifts the more frequently accessed product nearer the I/O point and shifts the less frequently accessed product away from the I/O point. Random and class-based storage represent two extreme points of the storage location assignment of ASRS. Random storage generally requires less storage space but class-based assignment reduces service time and increases throughput rates. Shuffling or relocation allows the combination of advantages from both extremes. This is accomplished by employing random storage location assignment but shuffling is initiated, when idle time occurs, to shift Class A pallets closer to the I/O points and Class B or C away from the I/O point. Two heuristics were developed to identify the shuffling route. The performance of the storage assignment policies were tested by using a simulation model written in SLAM II. This analysis shows that the SI-based shuffling (shuffling with insertion) is a better policy than SNN (shuffling with nearest neighbour) when the product turnover frequencies are not fixed. Shuffling during the idle time is rated as the better strategy to increase the ASRS operating efficiency.

According to Moon & Kim (2001) shuffling or relocations are helpful to maintain stable throughputs with all the three types of ASRS operation policies (random, 2 class-based and 3-class-based). They are also helpful to avoid losses caused by crane travel distance increase and lack of storage with a system under unstable production plans. Relocation does not cause any crane operation problems since the time to re-locate items in an ASRS is too minor to affect the crane utilisation. With class-based storage policies, better throughputs and lower rack and crane utilisations are achieved. An applicable operation policy can be selected based on the production plan variation, or a necessary variation point for relocation to the current policy can be determined using the simulation results.

Bozer & White (1990) have developed travel-time models for ASRS machines. The S/R machine is taken to travel simultaneously horizontally and vertically as it moves along a storage aisle. For randomised storage conditions expected travel times are determined for both single and dual command cycles. Alternative input/output locations are considered and various dwell-point strategies for the storage/retrieval machine are examined.

Chow (1986) developed a queuing model and a simulation model to study an AS/RS system performance under a first-come-first-served dispatching rule

Elsayed & Lee (1996) conclude that a nearest schedule batching rule (machine capacity not violated and tardiness considered) results in the smallest tardiness of retrieval and the shortest travel time. Other rules considered include: “Shortest processing time” (SPT) and “Most common locations” (MCL). Their findings indicate that the performance of these rules is not significantly affected by the S/R machine capacity and the “Modified Traffic Congestion Ratio” (MTCR) but is significantly affected by the order density.

According to Graves et al (1977) the worst scheduling policy presented is random storage assignment with no interleaving and first come first serve. Systems using this policy may have throughput dramatically increased in three ways: by adopting class-based storage assignment, by adopting mandatory interleaving, or both. They also conclude that class-based storage assignment has a cost: An increase of approximately 2% to 3% in rack size is needed for 2-class systems in order to keep the probability of default below 0.005; an increase of 4% to 5% is needed for 3-class systems.

Schwarz et al (1978) show with the aid of simulation models that the performance of a random storage assignment rule is equivalent to that of the closest open location rule. The system is also shown to be very sensitive to the utilisation or traffic intensity level as these levels approach one.

Their simulation also demonstrated that substantial increases in system throughput are obtainable from turnover-based storage assignment rules and from interleaving.

Elsayed & Unal (1989) argue that the optimum sequencing of order storage and retrieval is a complex problem. Another area highlighted is the effect of demand parameters (arrival rate of orders, probability distributions of order quantity and order locations) on the physical design of ASRS. They also suggest that researchers consider different cost factors such as inventory costs, operating cost of the ASRS and order delay costs when evaluating the effect of different inventory control policies on ASRS performance.

Egbelu and Wu (1993) report on the comparison of six dwell point specification strategies for S/R machines using average order turnaround time. When dedicated storage is employed a linear program 'LP Expect' (minimization of the S/R travel time) outperformed all other studied strategies.

Taboun and Bhole (1993) state that the performance of an ASRS is a function of both system configuration and item-pallet assignment. System configurations with mixed pallet sizes are superior in terms of system throughput (items/year) and other performance measures when compared to systems with a single standard pallet size.

Van Den Berg & Gademann (2000) report that on the basis of a simulation study the nearest-neighbour rule gives the best results for selecting an open location within the storage area for randomised storage or within a class-region for class-based storage. When an incoming load cannot be stored within its dedicated region it is better to assign it to a location further away from the input and output station, than to a location that is nearer than its dedicated region. The latter is likely to fill up the storage space for fast moving products, which may result in increased mean travel times. They also considered three criteria when evaluating good due date performance; mean response time, maximum response time or the number of late requests and report that these criteria were satisfied better when using a FCFS sequence for the retrievals than by applying specific urgency rules (giving priority to retrievals with long waiting times).

2.11 Knowledge Management

The concept of knowledge management (KM), which has received a great deal of attention in recent years, refers to the developing body of methods, tools, techniques and values through which organisations can acquire, develop, measure, distribute and provide a return on their intellectual assets [Kamara et al, 2002]. Within project-based industries the need for KM is fuelled by the need for innovation, improved business performance and client satisfaction within the dynamic and changing environment in which it operates. The fragmented nature of some industries means that efficiency in project delivery is less than desirable, resulting in dissatisfied clients and low profitability for many firms.

Knowledge is standardly defined as ‘justified true belief’ [Grayling, 2000] because at the very least it seems that to know something one must believe it, one’s belief must be true and one’s reason for believing it must be satisfactory in the light of some standard, because one could not be said to know something if one had, say, arbitrarily or haphazardly decided to believe it [Trought, 2001].

Another school of thought defines knowledge as belief: While belief is influenced by information processed by the believer, belief is not wholly influenced by processed information; there is room for insight, creativity and misconception [Fransman, 1994]. Lave (1993) stresses this with the observation that, ‘Knowledge undergoes construction and transformation in use’.

There appears to be two schools of thought – both apparent in management research – on how we actually accumulate knowledge. These are (a) the rationalist school, which holds that the chief route to knowledge is by the exercise of reason and logic and (b) the empiricist school, which holds that the chief route to knowledge is through perception [Grayling, 2000]. These two views are frequently in conflict. This conflict is generally considered to be caused by the human quality known as ‘consciousness’ which has been defined as ‘the specific nature of our subjective experience of the world’. A key subject of research in this area is subjectivity which may explain why workers and managers within manufacturing companies see things differently and hence so often come to different conclusions [Trought 2001].

2.12 The Value of Knowledge

To generate value, firms must be able to identify, create and continuously manage knowledge (especially technological knowledge) [Hitt et al, 2000]. Knowledge (that which is known and is a justified true belief) may be the most strategically significant resource a firm can possess and on which sustainable competitive advantages can be built [Marsh & Ranft, 1999]. Some scholars believe that competition is becoming more knowledge-based and that the sources of competitive advantage are shifting from physical assets to intellectual capabilities [Subramanium & Venkatraman, 1999]. Thus being able to develop, maintain or nurture and exploit competitive advantages depends on the firm's ability to create, diffuse and utilise knowledge throughout the company [Drucker, 2001].

The increasing competitive importance has led to the development of the knowledge-based view of the firm [Grant, 1996]. This evolved perspective, suggesting that the primary rationale for a firm's existence is to create , transfer and apply knowledge, is an extension of the resource-based view of the firm [DeCarolis & Deeds, 1999]. Spender & Grant (1996) argued that 'responding to the changes we see going on around us means bringing a better understanding of managerial and organisational knowledge and learning into a central place in the field's analyses and theories'. Sanchez & Heene (1997) maintain that organisational knowledge is the 'shared set of beliefs about causal relationships held by individuals within a group.

The ability to identify and especially to manage knowledge is the result of the firm's continuous effort to engage in learning. Because of the dynamic competitive landscape, advantage accrues to firms that are particularly adept at technological learning. Contextual factors that are either internal (e.g. firm size, structure, managerial ability) or external (e.g. industry) to the organisation may enhance or impede the firm's ability to engage in effective technological learning processes. Technological learning facilitates the firm's efforts to: (a) take appropriate levels of risks, (b) pro-act, (c) innovate, (d) develop, maintain and use dynamic core competencies, (e) build sustained competitive advantages and (f) create value. Evidence suggests that knowledge is central to how organisations learn and manage technologies [Oliveira, 1999]. An ability to understand and manage this relationship affects a firm's performance. Learning and knowledge are linked closely; knowledge is a critical outcome of learning. Beyond this, how knowledge is managed influences the selection and implementation of the firm's strategies [Teece et al., 1997].

2.13 The Relationship Between Technology and Knowledge

Among the many factors that will influence a firm's performance in the 21st century's competitive landscape, globalisation, technological advances and knowledge are perhaps the most significant [Hitt et al., 1999]. These three factors have both independent and interactive effects on the shape of the competitive landscape [Zahra et al., 1999]. Evidence suggests, for example, that in the biotechnology industry, technology and knowledge are highly inter-related [DeCarolis & Deeds, 1999]. Technology can be defined as a 'systematic body of knowledge about how natural and artificial things function and interact' [Itami & Numagami, 1992]. It follows that technology is a form of knowledge and that technological change can be understood by examining knowledge development [Bettis & Hitt, 1995]. Furthermore, as competition in global markets becomes driven more intensely and frequently by technology, technological knowledge may be even more important for firms with global ambitions [Boudreau et al., 1998].

2.13.1 Dimensions of Knowledge

There are different types of knowledge. The primary distinction among them is tacit knowledge and explicit knowledge [Polanyi, 1958]. The difference between these is described as the difference between experiential (i.e. tacit) knowledge and articulated (i.e. explicit) knowledge.

Tacit knowledge is accumulated through learning and experience; often, it is referred to as 'learning by doing' [Reed & DeFillippi, 1990]. Tacitness suggests that individuals know more than they can tell [Polanyi, 1967]. Tacit knowledge entails commitment and involvement in specific contexts and has a 'personal' quality [Nonaka, 1994]. As Polanyi (1958) stated, 'the aim of a skilful performance is achieved by the observance of a set of rules which are not known as such to the person following them'. Tacit knowledge is difficult to codify, articulate and communicate. Importantly, the tacit dimension does not suggest that knowledge cannot be codified. One view of this is that tacit knowledge may best be defined as knowledge that is not yet explained [Spender, 1996]. Terms such as 'know-how', 'subjective knowledge', 'personal knowledge' and 'procedural knowledge' have been used to describe the tacit dimension of knowledge.

In contrast to tacit knowledge, explicit knowledge can be formalised, codified and communicated. In fact, explicit knowledge is revealed by its communication while tacit knowledge is revealed through its application [Spender, 1996]. Concepts related to explicit knowledge include 'know-what', 'objective knowledge', 'pre-dispositional knowledge' and 'declarative knowledge'. The tacit and explicit dimensions of knowledge can reside in an individual or in the collective organisation. Spender (1996) incorporated these dimensions to develop a 2x2 matrix of organisational knowledge. The dimensions in his matrix (shown in Figure 2-1) are individual explicit (i.e. conscious knowledge), individual tacit (i.e. automatic knowledge), collective explicit (i.e. objectified knowledge) and collective tacit (i.e. collective knowledge).

A second distinction of knowledge types is between component and architectural knowledge. Component knowledge regards a particular aspect of an organisation's product, process or operation. Architectural knowledge, on the other hand, relates to the various ways in which the components are integrated and linked together into a complete system [Henderson & Clarke, 1990]. Thus, component knowledge can exist independently whereas architectural knowledge is embedded in a larger system and cannot be decomposed into independent parts [Garud & Nayyar, 1994].

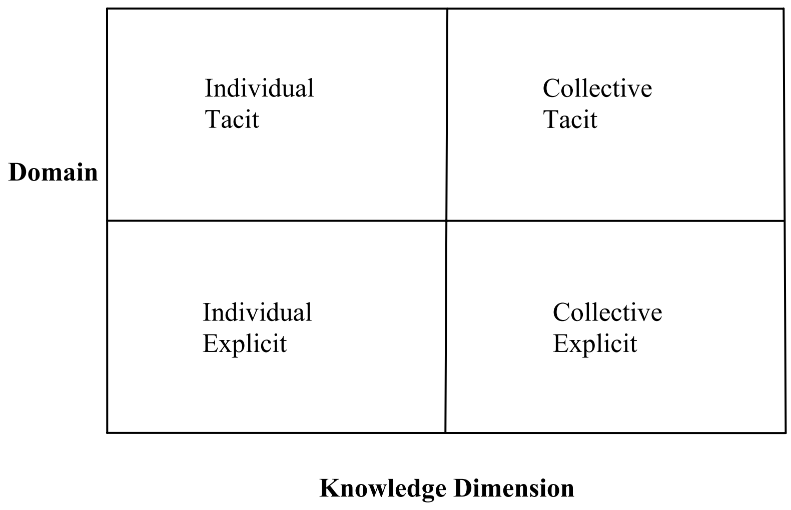


Figure 2-1: Organisational Knowledge Matrix

Many complex technologies can be described as a form of architectural innovation [Singh, 1997]. Component knowledge resides in either the individual or the collective and can be either tacit or explicit. However, because architectural knowledge is held throughout the whole organisation, it is collective in nature. Moreover, it is difficult for any one person to understand (or hold) the whole architectural knowledge, thus making such knowledge tacit by nature [Matusik & Hill, 1998].

2.13.2 Technological Knowledge

Technological knowledge (that is, knowledge that describes the functions and interactions of natural and artificial things) can be individual explicit (e.g. individual skills pertaining to a particular technology that can be codified), individual tacit (e.g. individual skills pertaining to a particular technology that is personal), collective explicit (e.g. standard operating procedures) or collective tacit (e.g. an organisation's routines and culture regarding technology). Each of these technological knowledge dimensions can be the source of competitive advantage and value creation [Spender, 1996]. However the dimensions that include a tacit component demonstrate the greatest potential for creating competitive advantages and firm value. Technological knowledge that is difficult to articulate, codify and explain is also difficult to imitate. Collective tacit knowledge is an important source of competitive advantage and value creation.

2.13.3 Developing Technological Knowledge Through Learning

Two outcomes result from the innovation process: innovation and learning. Acquiring technological knowledge (learning) is valuable because it leads to further innovation. Miller (1996) suggests that 'It remains unclear just what learning is, how it takes place, and when, where and why it occurs'.

The importance of learning has been examined at both the individual and organisational levels. Cyert & March (1963) stated that organisations are adaptively rational systems and that a theory of long-term behaviour in organisations must contain a theory of how organisations learn, unlearn and re-learn. Millar (1996) also states that 'organisational learning is the acquisition of new knowledge by actors who are able and willing to apply that knowledge in making decisions or influencing others in the organisation'. Thus learning entails acquisition of knowledge and in addition use of that knowledge in some way. These characteristics suggest two types of organisational learning, acquisitive and experimental. Acquisitive learning takes place as the firm acquires and internalises knowledge external to its boundaries. Experimental learning occurs largely inside the firm and generates new knowledge that is distinctive to the organisation [Lie et al, 1996]. On a relative basis, individuals and groups play a more active role in experimental learning than acquisitive learning. Through active experimentation and processes supporting it, individuals and groups 'learn' how to use organisational learning to create competitive advantages and value. Other learning typologies exist. One typology includes lower-level learning (single-loop learning or business-level learning), higher-level learning (double-loop learning or strategic learning) and meta-learning (incorporating a dynamic character). Lower-level learning involves developing rudimentary links between behaviour and outcomes through association building. It focuses on the immediate effect of the learning on some organisational task [Fiol & Lyles, 1985]. It is temporary and affects only a part of the organisation. Higher-level learning involves the use of heuristics and insights to develop 'frames of reference, interpretive schemes or new cognitive frameworks within which to make decisions'. As such, higher-level learning or double-loop learning takes place in complex and ambiguous situations. In general, lower-level learning is short-term orientated while higher-level learning is focused on the long-term. Although both learning types contribute to organisational success, higher-level learning is relatively more important when the firm seeks to create competitive advantage and value. As such, organisations must recognise and understand the set of factors that lead to higher-level learning.

Lei et al (1996) argue that learning helps build a firm's dynamic core competences. They suggest that firms can achieve higher-order learning based on three critical factors. The first relates to information transfer and retrieval that forms the foundation for a firm's universal and tacit knowledge base. The second concerns experimentation that allows firms to engage in continuous improvement and redefinition of heuristics. Finally, firms need to cultivate dynamic routines in order to develop firm-specific skills and capabilities thus achieving meta learning, which is defined as the simultaneous conceptualisation of different and contradictory forms of knowledge. Meta learning capability is especially critical for the firm seeking to define a new competitive space in uncertain, dynamic and volatile environments. A new competitive space typically is a product of proactive and innovative behaviour.

Three elements are required for an organisation to engage in successful meta learning. First, the firm must obtain explicit as well as tacit technological knowledge from internal as well as external sources. Second, the firm must engage in experimentation that results in continuous improvements. For organisations to survive and prosper, they must maintain a balance between exploration (i.e. experimentation with new alternatives) and exploitation (i.e. the refinement and extension of existing competencies). That is, companies must innovate and reap the benefits of that innovation. Finally, firms must build routines to effectively integrate technological knowledge throughout the organisation. This integration occurs primarily through sharing across individuals and groups.

Common knowledge is essential in developing dynamic routines. Common knowledge is known to all members of an organisation and enables people to share and integrate aspects of knowledge which are not common between them. Dynamic routines (i.e. the organisation's cognitive maps and particular approaches to framing) form the foundation for creating new technological knowledge [Lei et al., 1996]. New technological knowledge, in turn, can disrupt the status quo and thereby lead to innovation.

Dynamic routines are necessary for creative technological learning [Nelson and Winter, 1982]. Here, the role of an organisation's strategic leaders is critical. Strategic leaders must cultivate the necessary intellectual capital and create an environment in which innovation and knowledge are developed and exploited through continuous learning. Thus, the role of setting technology direction must come from the strategic apex of the organisation.

When a firm uses technological learning as the source of competitive advantage it cannot assume that its core competencies will remain valuable. Sudden and unpredictable changes in technological environments can alter the value of a firm's existing technological knowledge or render it obsolete. Therefore, firms must maintain a balance between cultivating core competencies as part of their knowledge-creation system while ensuring that the competencies do not become core rigidities. This is challenging because within each competence is the seed of rigidity. Organisational learning must be used to create dynamic core competencies. Dynamic core competencies may be grounded in either acquisitive or experimental organisational learning. However, it is unusual for a competence to endure for any period of time unless it is primarily a product of experimental learning [Lei et al, 1996].

2.14 Internal and External Technological Knowledge

Firms gain access to or form technological knowledge through two primary avenues. Experimental or internal technological knowledge is generated as individuals and groups experiment across multiple projects, including those involving research and development, manufacturing and marketing activities, Resulting from these experiments are unique and idiosyncratic insights about technological knowledge and its commercial application. Acquisitive or external technological knowledge is the knowledge gained and absorbed from sources outside the firm's boundaries. A source used increasingly for this purpose is inter-firm arrangements or collaborations. Both sources of organisational learning are important to the firm seeking to create competitive advantage and value.

2.14.1 Internal Technological Knowledge

Nonaka (1994) suggests that firms create knowledge through socialisation, combination, externalisation and internalisation processes. Developing technological knowledge is a product of the firm's 'transformative capacity', which is, 'the ability to continually redefine a product portfolio based on technological opportunities created within a firm' [Garud & Nayyar, 1994].

Kogut & Zander (1992) argue that a firm is a social community specialising in creating and transferring knowledge. They suggest that the advantage of the firm over the market does not lie in mitigating opportunism but rather in creating and transferring knowledge.

A primary internal source of technological knowledge creation is the firm's research and development program and capacity. Research by Cardinal & Hatfield (2000) shows that firms with separate research laboratories are more innovative than firms using a centralised R&D unit. Additionally, the location of these R&D laboratories was found to be important, in that laboratories close to the R&D center enhanced new product innovations as well as overall yields from basic research. However, corporate interference in these R&D laboratories had a negative effect on patent productivity. Another important internal source of technological knowledge is the innovation resulting from internal projects. March (1991) stated that exploration (i.e. behaviour entailing the search and experimentation with new alternatives) is critical for the organisation's survival.

Cheng & Van de Ven (1996) found that the innovation process is neither orderly nor random, but rather chaotic. They suggested that organisational learning can be seen as 'an expanding and diverging process of discovery'.

2.14.2 External Technological Knowledge

Cohen & Levinthal (1990) argue that a firm's 'absorptive capacity' is important. Absorbtive capacity refers to a firm's ability to recognise the value of new, external information, assimilate it and apply it to commercial ends. The ability to recognise, exploit and utilise external knowledge depends on a firm's level of prior-related knowledge. Thus, prior learning influences a firm's absorptive capacity. Additionally, to exploit the knowledge gained, it must be diffused to other units inside the organisation.

Firms often participate in multiple external networks and collaborate with other firms in multiple ways [Gulati, 1999]. The traditional reasons for collaborative efforts emphasised reducing risk and uncertainty. However, more recently learning has become an important motive (i.e. learning alliances) to acquire new knowledge. Kogut (1988) explains the rationale behind learning alliances, suggesting that firms can be thought of as knowledge bases and that joint ventures can be regarded as mechanisms to transfer tacit knowledge.

In addition to strategic learning alliances, inter-organisational networks are a major source of technological knowledge. For example many innovations occur outside of the company and even outside the industry that eventually commercialises them. Liebeskind et al. (1996) found that social networks helped new biotechnology firms obtain new technological knowledge as well as increase their scope of technological learning.

One form of external technological knowledge transfer is collaboration between firms and universities. Results show that knowledge transfer activities are facilitated when industrial firms have more mechanistic structures, cultures that are more stable and direction-oriented and when the firm is more trusting of its university research centre partner. While innovation creation is facilitated by organic structures, mechanistic structures may be superior for innovation implementation. Furthermore, a stable and direction-oriented culture imbues members with a common purpose and thereby reduces internal conflicts. Innovation creation and implementation create tensions in firms' cultures that must be managed effectively.

2.15 Managing Technological Knowledge

Despite its criticality, many firms lack the requisite capabilities to manage and make sense of technological knowledge. There are at least two reasons that organisations find it difficult to effectively manage technology. First, technology management is a complex process that encompasses R&D along with management of product, process and information technologies [Badawy, 1998]. Thus, technology management can be regarded as a 'dynamic system of interactions and inter-relationships between the sub-systems or sub-components of the organisation'. Second the very nature of the knowledge embedded in many technologies (intangible and tacit) makes it difficult to manage.

Two issues must be addressed before the firm can effectively manage technological knowledge. First, various knowledge integrating mechanisms must be in place to enhance the breadth, depth and speed of technological learning [Zahra et al., 1999]. Second, the firm must integrate technological knowledge with strategy in a dynamic manner. This integration requires the ‘architecture or configuration of management systems, policies and procedures governing the strategic and operational functional of the enterprise in order to achieve its goals and objectives’.

Efficient knowledge integration requires that firms develop the ability to access and utilise its employees’ specialised knowledge sets. This ability is determined by the level of common knowledge among the people in the organisation, how well organisational members communicate with each other and the organisation’s structure. The second characteristic is the ‘scope of integration’. In this context, scope refers to the breadth of specialised knowledge on which the organisation can draw. The greater its breadth of knowledge, the better a firm can develop complex technological products, in turn leading to ‘uncertain inimitability’. Finally firms need to access new knowledge or reconfigure existing knowledge to maintain competitive advantage and create value. These architectural innovations are linked closely to the stock of ‘architectural knowledge’ and how effectively firms engage in technological learning. Integration mechanisms are needed to bring together technological knowledge from within the firm as well as from external sources.

Itami & Numagami (1992) suggest that current strategy can help cultivate future technological knowledge and current technological knowledge can affect the cognitive processes of the managers who will form the future strategy.

Fowler et al (2000) suggest that long-term strategic success is created through strategies that focus on: (1) building market-driven, technological and integration competencies, and (2) decoupling these competencies from current products to create and exploit new opportunities. Such strategies then help firms develop dynamic competencies necessary in environments that are characterised by substantial change (e.g. the new competitive landscape). They suggest that strategic alliances may be used as vehicles to help firms integrate market-driven and technological capabilities. These embedded and often tacit competencies are critical for a firm to be strategically competitive (i.e. for the firm to create competitive advantages and value across time and events).

2.16 Summary

This chapter summarises the various components in an automated storage and retrieval system, listing also the benefits of automating a company's storage operation. Details of the various control strategies are included plus a summary of the performance measures applied to such systems. A thorough examination of current research topics in this area is also given. The findings from this review are that there is currently a large amount of research on-going with particular emphasis on improving throughput by analysing storage, retrieval and dwell point strategies.

The later section of this chapter concentrates on summarising the various aspects of knowledge management including: the value of knowledge, the relationship between technology and knowledge, internal and external technological knowledge and the management of technological knowledge.

The following chapter focuses on the ASRS design in terms of its mechanical and electrical configuration and a proposed control system.

3 ASRS Design and Build

3.1 Introduction

The ASRS developed in Waterford Institute of Technology is based on the Hi-bay storage facility in a large multi-national firm based local to the College. The reason for this correlation is to allow for concurrent research into ASRS control without incurring excessive downtime in their plant. This facility is currently controlled with Siemens S5 PLCs and custom-built warehouse management software (Harvest). It is intended in a planned upgrade to this Hi-Bay that the control hardware be upgraded to Siemens S7 PLCs utilising Step7^{TM} and WinCC^{TM} SCADA software with local communication to the company-wide ERP system (SAP).

The decision was therefore taken in the initial development stages of the W.I.T. ASRS that the hardware and software specified would match those selected by the multi-national for its Hi-Bay upgrade.

The proposed physical structure of the ASRS closely resembles the Hi-Bay structure of the partnership company. It consists of a single rack with 68 bays serviced by a crane with independently controlled horizontal, vertical and shuttle movements including external speed control. Within the rack are two bays dedicated to load and unload functions integrated with the existing FMS through forward / reverse conveyors.

The communications network proposed for the ASRS is a Profibus DP fieldbus network between the PLC, the remote I/O modules and the desktop PC. This PC operates under Windows 2000 with a WinCC function library DBExt.DLL, sitting between the Microsoft database and the ODBC compliant Siemens WinCC SCADA software. Siemens Step 7 software is used to program and control the Siemens S7-412-2 PLC. NetDDE.EXE is used to communicate between the WinCC based ASRS application and the FMS operating under Wonderware Corporation’s In Touch^{TM} SCADA in combination with G2^{TM} Artificial Intelligence, Microsoft SQL Server^{TM} and Visual Basic^{TM} interfaces.

This chapter will outline the steps involved in specifying the above hardware and software for this ASRS application and describes the physical build of the ASRS conducted chiefly by O’ Mahoney (2004) under the supervision of the author. It then goes on to describe the control, communication and interfacing of the ASRS that this research project was focussed on.

3.2 ASRS Mechanical and Electrical Design and Build

3.2.1 Introduction

This section of the thesis relates to the mechanical and electrical design and build of the ASRS conducted by O' Mahoney (2004) under the supervision of the author.

3.2.2 Storage Configurations

The ASRS design proposed was a unit load automated system designed to handle unit loads on specially designed pallets (Figure 3-1). Storage options in an ASRS consist of single, double and multiple deep. Double and multiple deep configurations are typically used to store identical items with loading and unloading at opposite sides of the racking to ensure correct product rotation (FIFO). Through examination of the product mix (Table 1) the decision was made to implement a single deep configuration. This configuration allows equal access to any stored component without obstruction and results in optimum retrieval times. It also allows for simplified simulation and comparison of results for a variety of retrieval and storage strategies.

Table 1: Components for Storage in ASRS

Presently	**Future**
Product 1: Components 1-9.	Tooling for robot.
Product 1: Clamps.	Probes for CMM.
Product 2: Components (Four Uprights and Eight Struts).	Product 3: Components.
Product 2: Jigs.	Product 3: Jigs and Fixtures.
Product 2: Clamps.	
Product 2: Final Assemblies.	
Various tool holders with fitted tools.	

The racking material specified in the design has an L-shape cross section. The material used in the construction was pre-drilled and pre-painted mild steel. Figure 3-2 shows a section of the racking. The storage capacity of the ASRS is equal to:
(Number of rows x Number of columns) – Unavailable bays
Resulting in:
Storage capacity = (6 x 12) – 4 (load & unload points require 2 bays each) = 68 bays
This number of bays is more than adequate for the product range in the FMS and is also ideal for testing the various control strategies outlined previously (Sections 2.4 – 2.9).

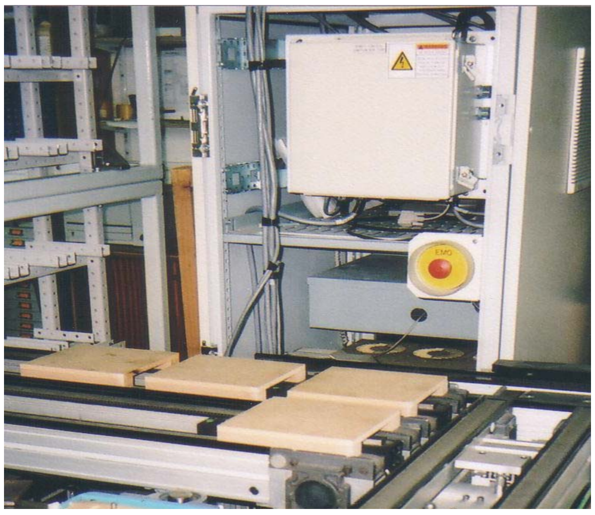


Figure 3-1: Pallet Design

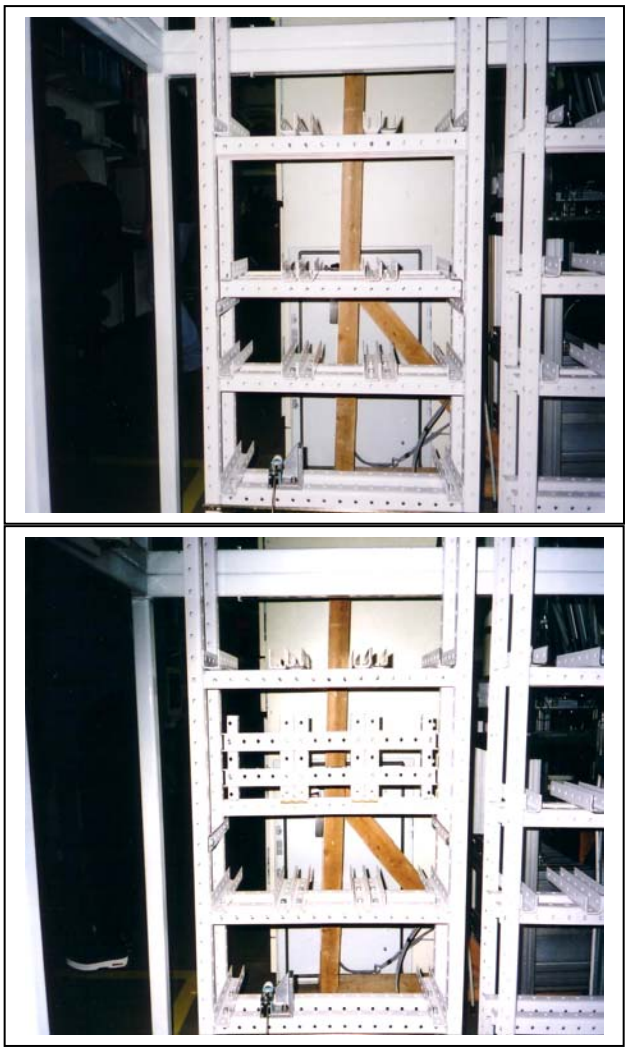


Figure 3-2: ASRS Racking

3.2.3 S/R Crane

The storage / retrieval crane requires three independent movements: horizontal, vertical and shuttle (forks forward / reverse).

A three-phase squirrel cage motor (Bosch NEMA Design B) was chosen for both the horizontal and vertical movement. These have good speed regulation combined with moderate starting torque.

Mitsubishi 500 series inverters were used (model FR-E540-0.4K) to control the speed, acceleration and deceleration of these motors. These inverters include a detachable interactive control keypad and display used for configuration and programming. The FR-E540-0.4K is a compact three-phase drive with high functionality (number of steps, pattern, external control options), powerful overload protection, parameter copy and update capability, low audible motor noise and compatibility with MX500 set-up and programming software. It has a capacity of 0.4 KW and an output current of 1.6A.

The S/R machine consists of a mast that spans the rack from top to bottom, this mast has at either end a base and a top assembly (Figures 3-3, 3-4 and 3-5 show the initial crane design [O' Mahoney, 2004]). The base assembly consists of a three-phase induction motor for horizontal movement plus 2 pairs of wheels which guide the unit along a horizontal track, restricting any lateral movement (Figure 3-6). This base assembly also contains a drive drum (Figure 3-7) which gathers and releases the steel cable used for vertical movement. A top plate assembly was mounted at the mast top (Figure 3-8). This also contains two pairs of wheels which run either side of a horizontal track at the top of the ASRS. The top assembly has only a guide and support function. There is no drive requirement at this position with the top horizontal guide beam also performing a support function.

The vertical movement of the S/R machine is achieved by directing the steel cable from the drum on the base assembly through guide pulleys on the top assembly and attaching it to a vertical unit which runs on the vertical mast.

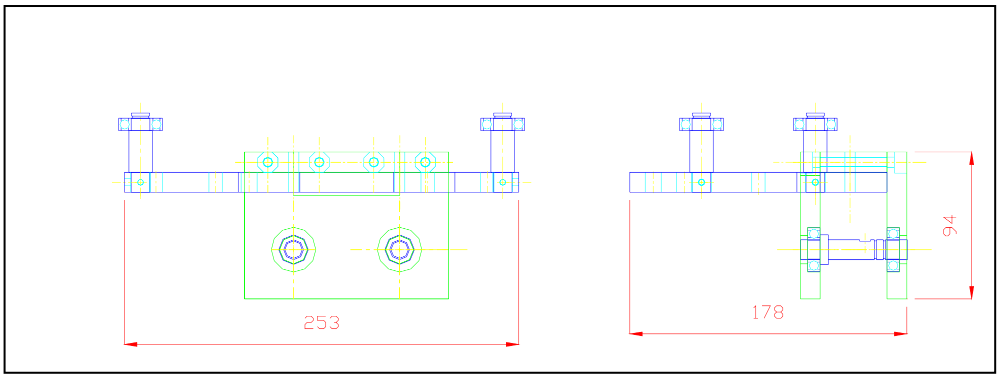


Figure 3-3: Top Plate Design

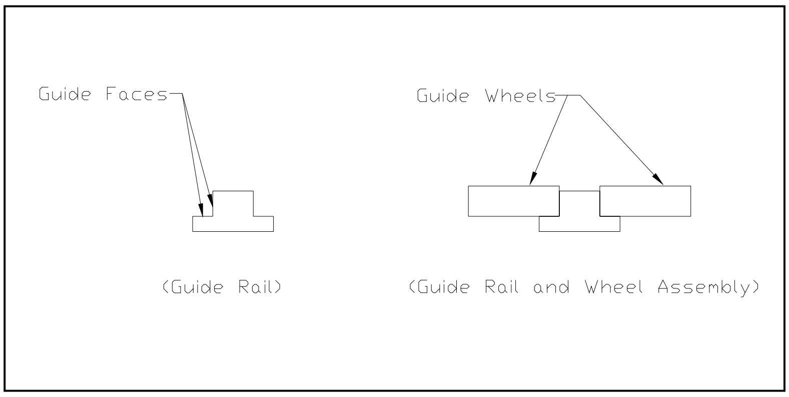


Figure 3-4: Guide rail and wheel conceptual design

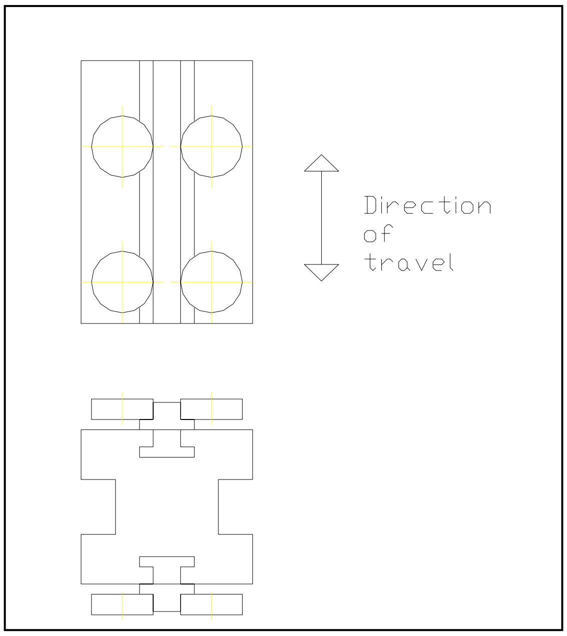


Figure 3-5: Guide Rail Assembly Conceptual Design

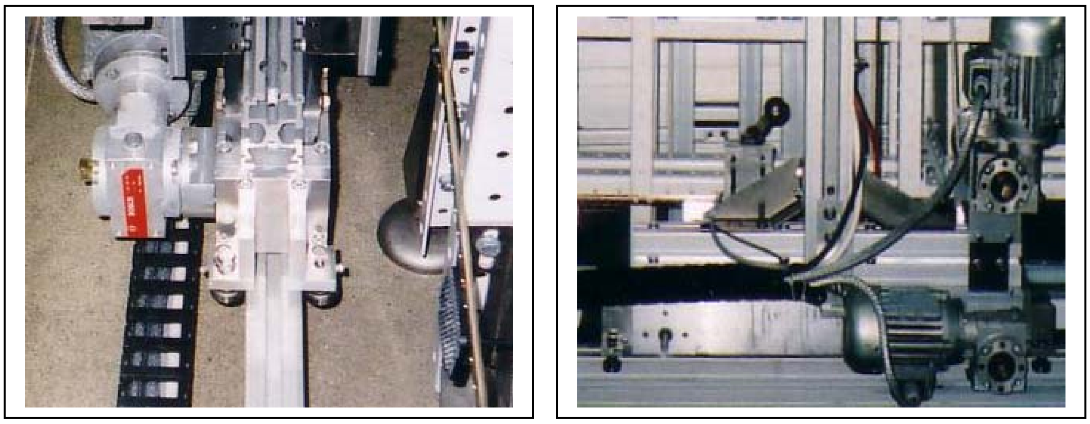


Figure 3-6: Base Assembly

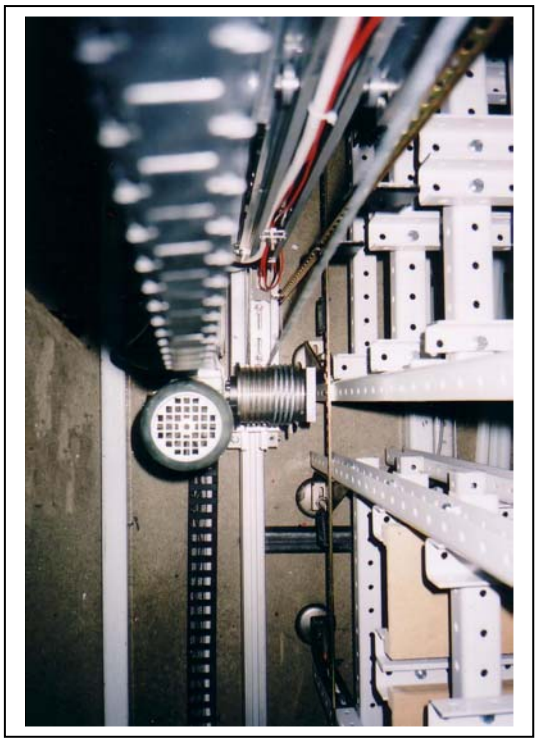


Figure 3-7: Vertical Drive Drum

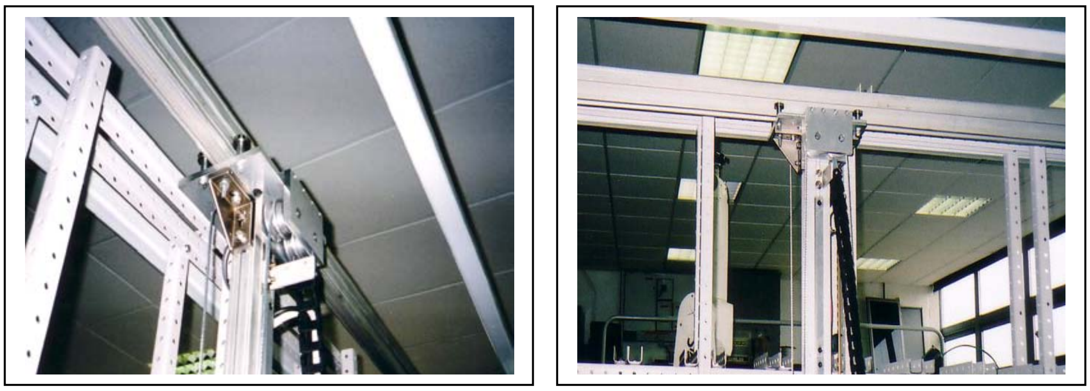


Figure 3-8: Top Plate Assembly

3.2.4 Shuttle Movement

The shuttle design consists of a single DC motor driving a telescopic fork set that has three individual chains for drive transmission. When the S/R machine reaches the desired location the single DC motor is activated.

A Bodine model 24A4BEPM-D3 DC geared motor was chosen for this application. This motor has a peak torque of 4.5 Nm and a horsepower rating of 0.044 kW. It is controlled using a Minarik XL3025 pulse-width modulated adjustable speed DC drive controller.

The shuttle forks mechanism consists of three plates and three drive chains which allow for extension of the forks in either of two directions, this allows for further expansion of the ASRS to include a second rack which can also be serviced by this one S/R machine. Figure 3-9 shows the shuttle fork mechanism together with the Bodine model 24A4BEPM-D3 geared DC motor.

Figure 3-10 illustrates the wiring arrangement for the speed control of the forks. The Live and Neutral of the 220 VAC supply are connected to L1 and L2 on the controller terminal block. A speed adjusting potentiometer is connected to terminals S1, S2 and S3. Terminals A1 and A2 are wired through a series of Allen Bradley four pole double-throw relays for hardwired interlocking of forward and reverse movement.

Figure 3-11 shows the main control cabinet which includes the Siemens PLC, the two Mitsubishi inverters, the Minarik DC controller and a number of single and three pole contactors, MCBs and relays.

3.2.5 Electrical Supply

A Residual Current Device (RCD) protects the main power supply to the AMT Laboratory. The 3-phase power supply to the ASRS control cabinet is isolated with a wall mounted isolator switch plus an isolator switch mounted internally in the cabinet. Phase power is wired directly from this isolator switch to the Siemens S7 PLC and the 24 VDC power supply. Both of the Mitsubishi inverters and the single Minarik DC drive controller are wired through a contactor which is powered through the emergency stop circuit. Activation of any one of the emergency stops (Figure 3-12) will therefore result in power to the motor inverters and controller being shut off, power to the PLC and 24 VDC power supply will remain on. To reactivate power to the inverters and DC motor controller, the emergency stops must be depressed and a restart button pressed.

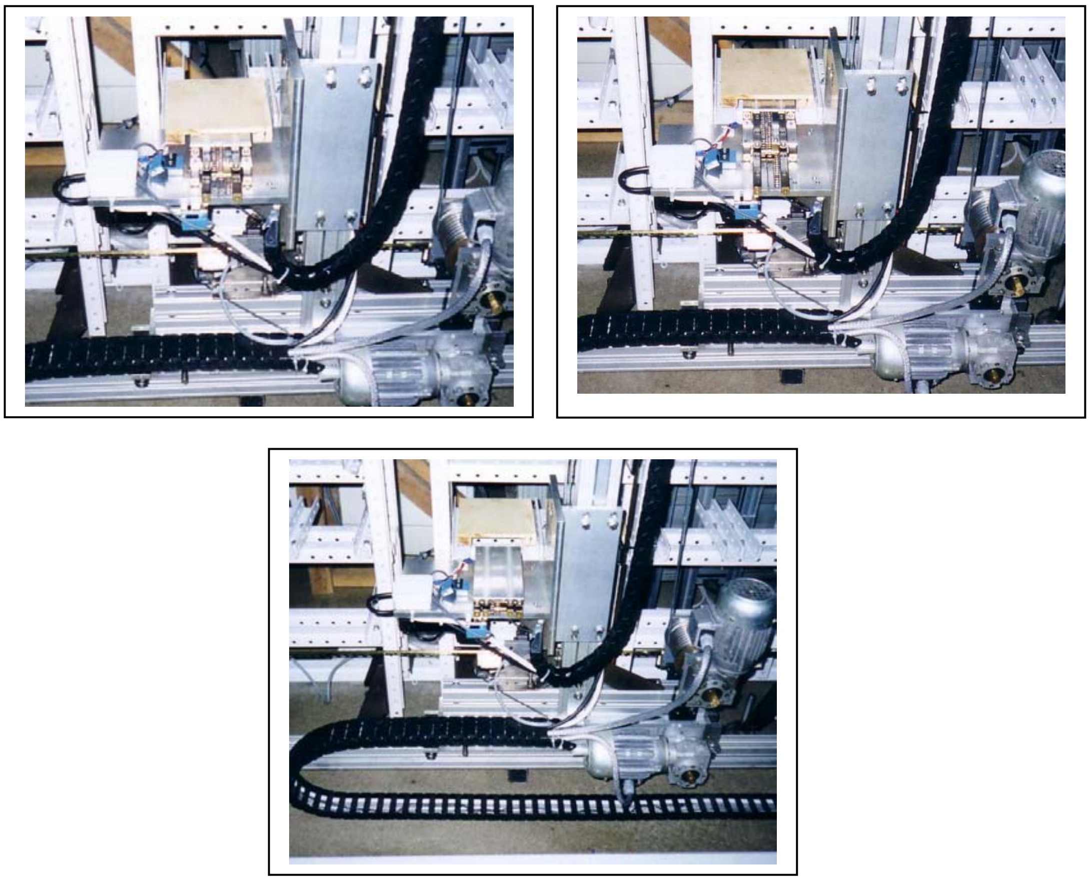


Figure 3-9: Shuttle Fork Mechanism

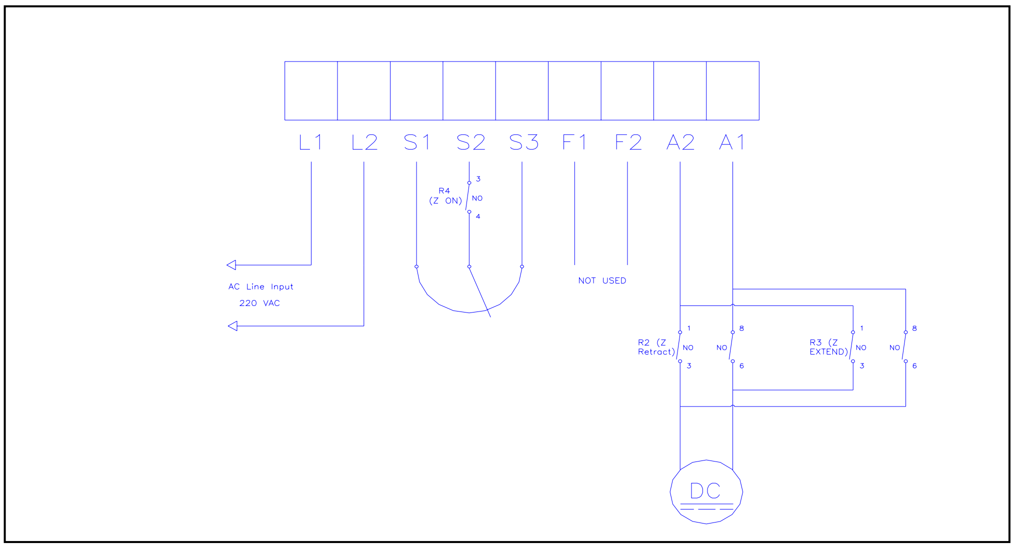


Figure 3-10: Shuttle Forks Speed Control

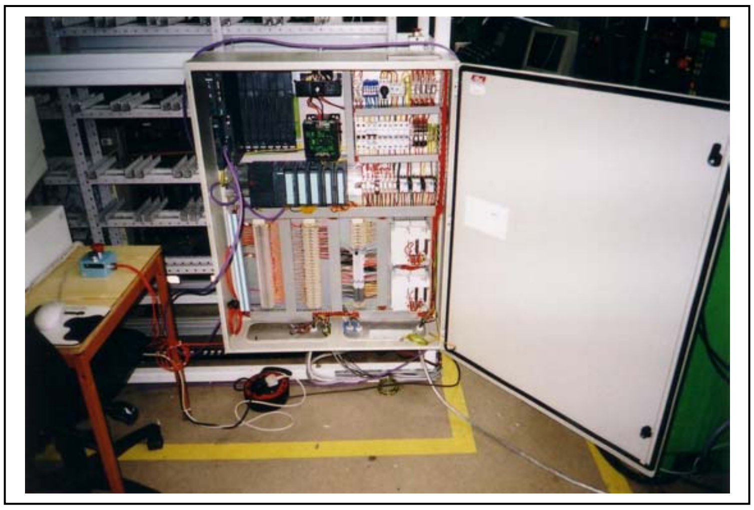


Figure 3-11: ASRS Control Cabinet

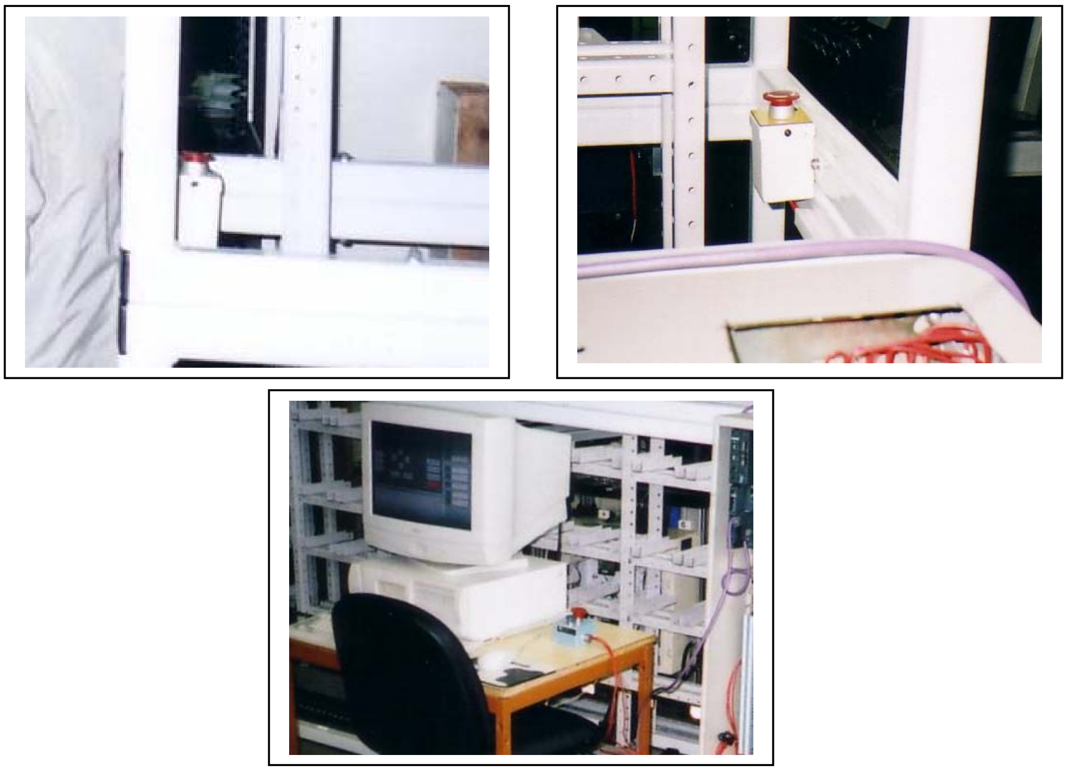


Figure 3-12: Emergency Stops and Start/Reset Switches

3.3 ASRS Control, Communications and Interfacing

3.3.1 Introduction

This section of the thesis relates to the selection of the ASRS control hardware and software including the PLC and programming software, SCADA software, database selection and design and the development of an order generator. The integration of the ASRS with the existing FMS is also considered.

The control of this existing FMS is through an intelligent scheduler and dispatcher developed by Flanagan (2004), communicating with a MS SQL Server database over a NetDDE communications network with Visual Basic interfaces to individual machines. Current research by Roche (pending) is on the generation of a business development interface which will link the existing system to the customer through the use of an order generator hence allowing for a thorough testing of the scheduler.

3.3.2 PLC Selection

The make and model of the PLC chosen for this application was influenced firstly by the decision of the partnership organisation to remain faithful to the PLC manufacturer of their existing system and secondly by their decision to include in the upgrade specification the use of Profibus as the networking choice.

It was therefore necessary to select a PLC from the Siemens S7 range for the WIT ASRS which also had to be Profibus compliant. The S7-200 range does not include Profibus masters, the S7-300 range has Profibus masters in the higher models, from the 315-2 DP model to the 318-2 DP model. The S7-400 series also includes Profibus DP masters in its range.

The model chosen by the Partnership organisation to replace their two existing S5-150U PLCs was the S7-417-2. This model offers a programming memory space of 1.6 MB and an execution time of 0.08 μs. The specification of the PLC for the WIT ASRS was not quite so demanding on required programming memory and peak execution time although it was essential that it could perform as a Profibus DP master. The S7-300 has a totally different rack type than the S7-400 and for this reason was not considered. From the S7-400 series the model chosen was the S7-412-2 which contains 144 KB of programming memory combined with 0.2 μs execution time.

3.3.3 Local Communication

The proposed communication between the PLC, I/O modules and PC is with Profibus DP which allows for the connection of field devices such as I/O modules with automation systems.

The master PLC and the distributed I/Os both require Profibus ports. This is already ensured in the specification of the S7-412-2 master PLC. Profibus allows for quick disconnects, easier wiring and efficiency benefits when equipment is being disassembled and reinstalled.

Siemens claim that the installation of twisted-pair fieldbus in material handling / conveyor applications generates savings of 50-60% or more over hardwiring all the input and output devices directly to the PLC terminals.

Using Profibus, data from field devices can be used to prioritise maintenance jobs. Instead of routinely pulling and diagnosing components, which may catch them before maintenance is needed, or after performance degrades significantly, users can now replace components using data from the devices themselves, which can greatly reduce costs and downtime.

According to Siemens, Profibus also offers savings in wire, terminations and labelling and associated labour costs.

3.3.4 PLC Software

The PLC programming software chosen for this design is Siemens Step 7. This program, Siemens claim, allows for easy creation, testing, start-up, operation and maintenance of control solutions through the provision of an integrated tool set for all system components. The software stores all configuration and programming data in a central database to which all the modules have access. Data is required to be entered only once and is then available for the entire project.

The Simatic Manager is a graphic user interface which allows configuration and offline/online editing of S7 objects (projects, user program files, blocks, tools and hardware stations). Simatic Manager enables the user to:

- Access the PLC online
- Manage libraries and projects
- Use Step 7 tools
- Edit memory card programs

With Simatic Manager data is stored in the form of objects in a project. These are stored in a hierarchical structure (project tree). The project tree consists of the following levels:

Level 1: The project icon represents a database where all the relevant project data are stored.
Level 2: Stations where hardware configurations and parameter assignments are stored with program folders containing program blocks and sources for writing Step 7 PLC programs and overall network components such as MPI/Profibus/Industrial Ethernet

Step 7 provides various types of blocks in which programs and related data can be programmed and stored. The structure of the program will differ with the process requirements and will consist of some of the following block types:

Organisational Blocks (OB): These provide the interface between the user program and the operating system. The OS cyclically calls OB1, which can contain the entire program or calls to other sections of code (functions, function blocks or data blocks).

Function (FC, SFC): A function contains a piece of functionality of the user-defined program and does not require access to data blocks. A system function has a fixed functionality and is integral to the CPU's operating system (time functions, block transfer)

Function Block (FB, SFB): Function Blocks and System Function Blocks are similar to Functions and System Functions except they have their own memory area in the form of instance data blocks.

Data Blocks (DB): Data Blocks are data storage areas for storage of user data.

3.3.5 HMI Software

To ensure full compatibility with the Siemens S7 PLC and to model the Partnership organisations upgrade the HMI software proposed is Siemens WinCC.

This software is designed for visualisation and operation of processes, production flows, machines and plants. The base system, which is industry and technology independent, is suitable for universal use in any automation application according to Siemens.
WinCC offers simultaneous access to a maximum of six WinCC process data servers and the representation of all associated information. It also supports multiple users and Siemens claim that it provides excellent visualisation and communication functionality.
According to Siemens it has powerful process interfaces, in particular to the Simatic range including Ethernet / Profibus / AS-i and open communications through OPC / OLE / DDE and ODBC. The basic package is integrated with Simatic Step 7 for ease of configuration and includes functions for signalling events, archiving of measured values, logging of process and configuration data, user administration and visualisation. The application is designed for visualisation and operation of processes, production flows, machines and plant. It offers easy to install client server structures and simultaneous multi-client access of up to six WinCC process data servers. WinCC version 5 enables process management via the Intranet / Internet using Web Navigator with no additional configuration required.

3.3.6 Database Design

Recording of the storage details of the ASRS is required at a central location as the future development of the laboratory requires easy access to all records. Currently the FMS data is stored in an MS SQL Server database using an ODBC connection for communication to the control system. Full integration of the ASRS into the FMS requires communication and data storage at a level comparable with the FMS control components. This means that the main control FMS database will also contain tables of data on the ASRS such as contents, classification of bay groupings, allocation of dedicated storage to bays, requirements etc. These tables can then be checked by the MRP system to determine the status of various components prior to the launching of new works orders.
For the current research project a decision was made, for practical reasons of autonomy, to opt for a local MS Access database on the ASRS PC with communication through a pre-defined ODBC link to the ASRS control programs. It is

envisaged that this will be altered at a future date to link the control software to the centrally located MS SQL Server Database for increased security and access control. To handle the communication between the WinCC SCADA software and both local and networked SQL applications it is proposed to use the ODBC interface provided with Windows 2000. This assumes that the applications and database are ODBC compliant. WinCC, which is ODBC compliant, combined with a communications add-on (DBExt.DLL) allows for this.

DBExt.DLL consists of a set of synchronous and asynchronous functions for reading and writing of database records. Synchronous functions wait for a result using time intensive requests and block global scripting during connection problems. Asynchronous functions return immediately using a call-back function allowing global scripting to continue and are characterised by the addition of 'CB' at the end of the function name. DBExt.DLL is a library of database functions, realised as a DLL and it displays record fields using WinCC standard I/O objects. It has a wide range of functions to allow for easy and efficient communication between the application and the database.

3.3.7 ASRS Order Generator

In order to conduct a complete set of tests for pathological conditions on the physical and mathematical ASRS models it is necessary to create an order generator. This program has to have the capability of generating a range of orders which covers all possible conditions for both models including the extremes of requirement type, demand interval and component type. Such a demand generator will test the ASRS models to an infinitely greater degree than any set of orders that are developed manually.

The system specified is a SIMUL8 model which will allow the user to specify: the list of components, the required operation, the probability of time interval between jobs and the total number of jobs required. This model will transfer the generated list of orders (excluding the time interval) to a temporary Requirements table in a MS Access database, the time interval values will be transferred to a separate table in the same database. These two tables will be combined using a macro within the MS Access database that asks the user for a "Start time" during its operation and outputs the new list of requirements to a Requirements table. This will allow for the re-running of the same Requirements by the ASRS control system, with a different start

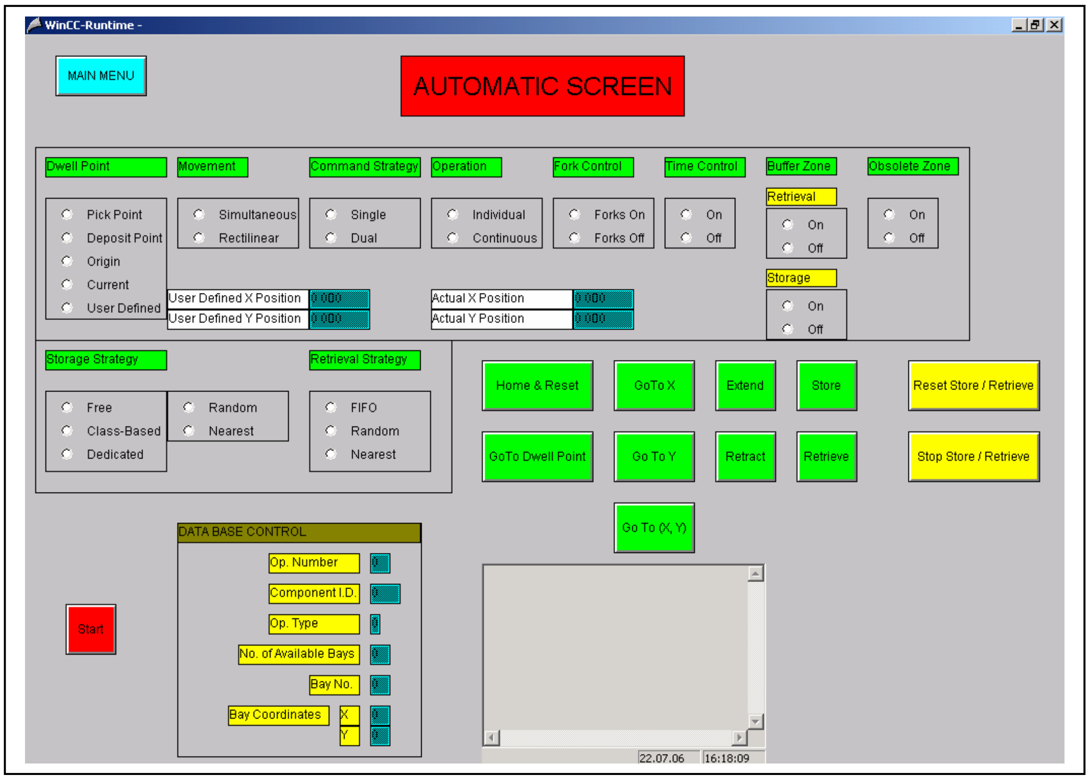
time, without having to re-run the order generator, ensuring the operator can run a number of replication trials on the same data.
The proposed mathematical model will have the capability of importing these requirements from the database and then simulating the actual run data. This will aid the researcher in building the required confidence in both models.

3.3.8 ASRS / FMS communication and Interfacing
The control of this existing FMS is through an intelligent scheduler and dispatcher developed by Flanagan (2004), communicating with a MS SQL Server database over a NetDDE communications network combined with custom applications which provide interfaces to individual machines.
Future development of the ASRS will concentrate on a complete integration with the FMS. This will incorporate the development of new tables within the control MS SQL Server database and the establishment of a communications link between the PCs. This link will be across a NetDDE network similar to the existing FMS control network. The planning and control of the FMAS is managed by an MRP/ERP system which requires this link for it to consider raw materials or finished goods stored in the ASRS. The scope of this research did not extend to the development of this link. This really is unacceptable from a planning perspective and will need to be addressed as a matter of urgency in the on-going development of this system.

3.4 System Design

3.4.1 Introduction
This section of the thesis relates to the design of the PLC and SCADA control programs to ensure that the system complies with the overall project objectives of designing an ASRS test rig that is capable of being used as a tool to investigate a range of ASRS characteristics.

3.4.2 SCADA Programming
The proposed design (Figure 3-13) comprises of a master WinCC HMI screen which will allow for the setting of a number of ASRS characteristics including: Command strategy, dwell point location, travel paths, storage strategy, retrieval strategy, single or multi-product types, bay availability, pick and deposit locations, obsolete strategy and buffer zones (for pre-retrieval and pre-storage)



User Defined X Position 0.000
User Defined Y Position 0.000
Actual X Position 0.000
Actual Y Position 0.000

Op. Number 0
Component I.D. 0
Op. Type 0
No. of Available Bays 0
Bay No. 0
Bay Coordinates X 0
Y 0

22.07.06 16:18:09

Figure 3-13: Proposed Control Screen

<u>Command strategy</u>

This gives the operator the choice between single or dual cycle, this may not always be significant because at times there may be only a requirement for one type of operation, i.e. a series of retrievals required, especially at start-up but it is essential for the analysis of ASRS performance.

<u>Dwell point location</u>

The proposed choices available to the operator are:

Pick Point

Deposit Point

Current Location

Origin (0,0)

User Defined

These options are necessary both to match current research into cycle time analysis and also to conduct future research into dwell point locations.

Travel paths

The proposed options for this parameter are:

Rectilinear

Simultaneous

There is a large amount of on-going research whose results are based on rectilinear travel, therefore it is important to include this control option. In general operation, it is more appropriate to use simultaneous travel (faster, more efficient).

Storage strategy

The six options proposed are:

Free – Random

Free - Nearest

Class-based – Random

Class-based - Nearest

Dedicated - Random

Dedicated - Nearest

The majority of the research into cycle time specifies random storage, however there is a significant body of research into other strategies. The proposal aims to develop a control system that can be used to evaluate these various storage strategies. The selection of class-based storage will be dependent on whether there is single or multi product types requiring storage. The system proposed will have the capability of categorising various sections of the racking for the required classes or of allocating particular bays to particular products (dedicated).

Retrieval strategy

The options proposed here are:

Random

Nearest

FIFO

The most common strategy in current research is random selection of the products for retrieval. The most common in practice is FIFO to ensure reasonable stock rotation.

Bay availability

For a variety of reasons (restricted access, damage, ease of access) there may be certain bays in the racking to which an operator may want to restrict access. A database linked to WinCC will control access to the bays opening and closing access whenever it is required.

Pick and deposit point locations

In the proposed ASRS these locations are defined by the conveyors to and from the FMS. In other circumstances it may be necessary to either transpose these locations, use one conveyor for both pick and deposit etc., flexibility is therefore required in the specification of these locations. It may also be required to specify a completely different location for pick and / or deposit.

Obsolete strategy

The option will be available to turn on or off this function. When this strategy is selected there will be a zone of the racking allocated for obsolete items i.e. items not required in a defined time period. These items will be moved to this zone during S/R slack periods. Reports will be available on the contents of this zone and their movement history. A further option will be available to empty the obsolete zone if necessary.

Buffer zones

There is proposed to be three options:
Pre-retrieval buffer zone
Pre-storage buffer zone
No buffer zone

The operator can select either option. The pre-retrieval buffer zone is used for ‘kitting’ of components and / or parts prior to delivery to the FMS. This will theoretically result in improved cycle times by taking advantage of slack time to query the work order requirements and re-organising product within the ASRS (Shuffling or Relocation). The Pre-storage buffer is used for rapid storage, if there are a number of pallets to be stored in a short period then this option will ensure that the travel distances for the pallets will be as short as possible, thus improving the ASRS reaction time.

Neither the Obsolete or the Buffer Zone control was developed as part of this research program. It is planned to develop these in the near future.

Manual control

One additional feature of the WinCC control which is envisaged is the inclusion of manual control. This will allow the operator to manually select and de-select the following: S/R machine Up, Down, Forward, Reverse, Forks In and Forks Out, GoTo X (horizontal travel), GoTo Y (vertical travel), GoTo (X,Y), GoTo Dwell Point, Home and Reset.

3.4.3 PLC Programming

The proposed Step 7 program will adhere to a structured format as summarised in Table 2.

The major benefits of adhering to such a rigid structure are:

- Ease of programming
- Efficient fault-finding
- Safe operation
- User-friendly upgrading
- Isolation of program elements

These functions are called from OB1 which allows for complete isolation of the different strategies and control options. This is ideal for a student learning environment and also for the on-going development of the ASRS control to analyse various new and improved strategies. The PLC programming is detailed in Chapter 4 where a number of techniques used to implement this control design are highlighted.

Table 2: Proposed PLC Blocks

Program item	**Description**
OB 1	Main Program Control
FC 1	Manual control, X and Y axes
FC 2	5 Second Time Delay
FC 14	Home & Reset
FC 15	GoTo X
FC 16	GoTo Y
FC 17	GoTo (X,Y) – Simultaneous Movement
FC 18	GoTo (X,Y) – Rectilinear Movement
FC 19	Pick – Time Delay
FC 20	Place – Time Delay
FC 21	Semi-Automatic Pick
FC 22	Semi-Automatic Place
FC 23	Automatic Pick
FC 24	Automatic Place

3.5 Summary

This chapter details the proposed design and build of the test-bed ASRS for W.I.T. including the mechanical and electrical components plus the communications, control software and programming requirements.

The main emphasis of the hardware selection process was to ensure reasonable compatibility with the partnership organisations proposed up-grade. This is achieved through the design of the racking and S/R machine plus the selection of Siemens control hardware and software.

The flexibility in control of the ASRS is important from a research stand-point. It is important for W.I.T. that this ASRS should be capable of a variety of control strategies to analyse the impact of these strategies on the overall performance of this ASRS and furthermore to develop theories on a variety of control strategies including: command strategy, dwell point location, travel paths, storage and retrieval strategies, obsolete and buffer zone implementation and possibly others into the future. This chapter highlights the control process to be implemented and summarises the various programming elements that will be developed.

The next step is to outline the development of the control system for this test-bed.

4 ASRS Control and Communications

4.1 Introduction

This chapter summarises the programming and control of the ASRS in the AMT laboratory in Waterford Institute of Technology.

Profibus DP, DBExt.DLL, NetDDE are specified for the communications network to support this ASRS and to link it to the existing FMS control within the AMT space. Siemens Step7 and WinCC are the proposed PLC programming and HMI software controlling a Siemens S7-412-2 PLC.

The control of the ASRS includes an inbuilt flexibility to enable the testing of a number of different control strategies, including: storage, retrieval, command, dwell point, travel type and buffer and obsolete zone implementation. The programming, in Step7 and WinCC, to achieve this control is described in this chapter. Also included is a description of the control database and the order generator.

The linking of the ASRS to the existing FMS was not developed within this research program.

4.2 Development

There have been a number of significant steps in the development of the control programs for this test-bed. These have included:

- Initial programming of specific functions by O' Mahoney (2004)
- Specific strategy related PLC functions
- Manual control of S/R machine and forks
- Button activated SCADA Scripts
- SCADA and PLC programs for specific requirements and locations
- Generic PLC functions
- Integration of database communication into SCADA programming
- Development of SCADA project functions and actions
- SCADA and PLC programs for a variety of strategies
- Integration of more advanced strategies into script
- Integration of project functions and global actions in SCADA script
- Development of database to ensure the re-evaluation of trial data
- Development of Order Generator
- Development of mathematical model

- Trials
- Timing of calculations changed in data control
- Further trials

The initial work completed by O' Mahoney (2004) under the supervision of the author concentrated on basic PLC functions and button activated SCADA script to perform very basic control. Through a series of projects supervised by the author these basic functions progressed to the development of ASRS strategies including command, travel type and dwell point with knowledge being continually expanded during this period.

A major event during the course of this research was the development of generic PLC functions. One such function (labelled FC 17 in the PLC program) controls the movement of the S/R machine to a required destination. This particular function is called on numerous occasions and from a variety of scripts to perform the same function, i.e. to move the S/R machine to a specified location.

Another progressive step in the development of this test-bed was the integration of database communication utilising the WinCC function library DBExt.DLL, with SCADA project functions and then with SCADA global actions. This broadened the scope of the work and allowed for the loading and completion of a range of requirements.

Further development then concentrated on the programming of scripts to control the various strategies selected and the development of an order generator communicating with the MS Access database. This work allowed for the trialling of the test-bed performing under the various strategies and a comparison of these results with those obtained from the ASRS mathematical model developed in SIMUL8. The lack of correlation between these sets of results led to further refinement of the physical model in the general area of data flow and control between the project functions, global actions and the PLC functions.

The correlation between the two models has now been proven for a wide variety of strategies and future research can concentrate on the development of further strategies in the physical model and on long term trials on the mathematical model using its own order generator.

The dispersion and creation of knowledge has been paramount during this development and this is outlined in Chapter 6 as a case study in Knowledge Creation based on a specific model proposed by Nonaka et al. (1994). This chapter concentrates on the description of the current status of the test-bed highlighting some of the aspects of the various programs integrated into the system.

4.3 Control

4.3.1 Introduction

This section summarises the programming of the ASRS which consists of the following software: SiemensStep7, SiemensWinCC, MSAccess, SiemensDBExt.DLL and SIMUL8.

4.3.2 Control Sequence

A flow control diagram for the ASRS is shown below in Figure 4-1. Control consists of PC based code in the form of WinCC Global Script Project Functions and Actions and PLC Step 7 functions. One important feature of the state diagram is the timing of the calculations to determine the co-ordinates of the required bay. During the initial trials on this system it was noticed that there was a slight pause in the S/R machine movement at a number of stages in the cycle as the system waited on a decision from the Windows 2000 based SCADA control. This was overcome by altering the sequence of these calculations so that they were concurrent with a PLC operation. This added a large degree of complexity to the control, especially for Dual commands and continuous operation.

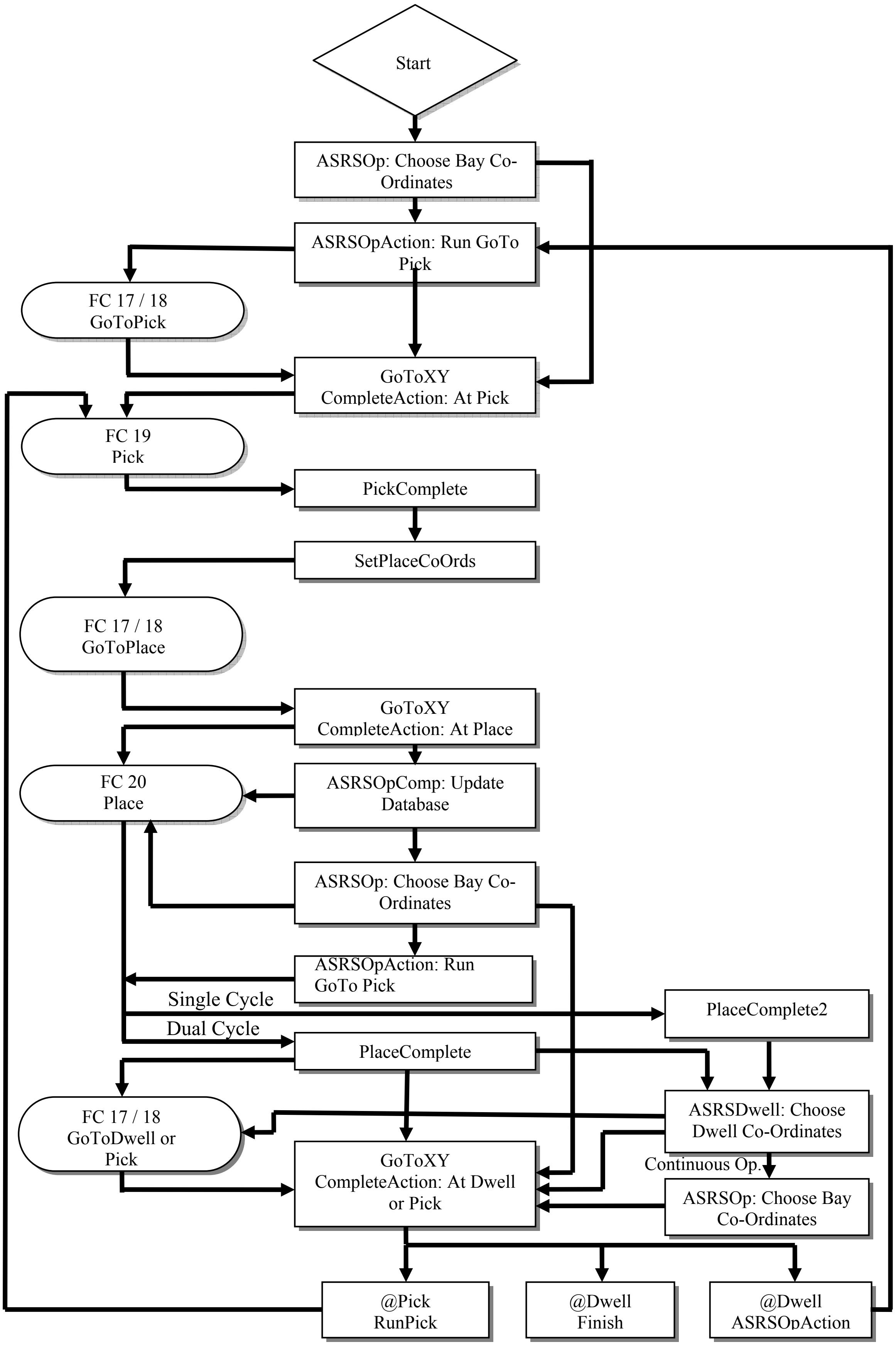

Start
ASRSOp: Choose Bay Co-Ordinates
ASRSOpAction: Run GoTo Pick
FC 17 / 18 GoToPick
GoToXY CompleteAction: At Pick
FC 19 Pick
PickComplete
SetPlaceCoOrds
FC 17 / 18 GoToPlace
GoToXY CompleteAction: At Place
FC 20 Place
ASRSOpComp: Update Database
ASRSOp: Choose Bay Co-Ordinates
ASRSOpAction: Run GoTo Pick
Single Cycle
Dual Cycle
PlaceComplete2
PlaceComplete
FC 17 / 18 GoToDwell or Pick
ASRSDwell: Choose Dwell Co-Ordinates
Continuous Op.
GoToXY CompleteAction: At Dwell or Pick
ASRSOp: Choose Bay Co-Ordinates
@Pick RunPick
@Dwell Finish
@Dwell ASRSOpAction

Figure 4-1: Flow Control Diagram

4.3.3 Control Interfaces

The WinCC SCADA package can be summarised in three distinct categories with a large degree of interaction between these. The three categories are:

WinCC User Interface Screens

WinCC Project Functions

WinCC Global Actions

These will now be discussed in detail with thought also being given to their interaction.

4.3.3.1 User Interface Screens

The graphics user interface of the ASRS consists of five WinCC screens developed in its Graphics Designer:

- Main Menu
- Semi / Full Automatic Screen
- ASRS layout and Contents
- Class-based Storage
- Dedicated Storage

<u>Main Menu</u>

This is an introductory screen to the ASRS control. It consists of a series of control buttons that allow access to each of the other four screens. It also allows for de-activation of the WinCC runtime mode. This screen, shown in Figure 4-2, is the default screen for the ASRS and is the only screen from which the other screens can be accessed.

<u>Semi / Full Automatic Screen</u>

This screen, shown in Figure 4-3, controls the operation of the ASRS. It requires the operator to pre-select the various strategies to be employed during automatic operation including: Dwell Point, Travel Type, Command Type, Operation Type, Time Control, Fork Control, Obsolete and Buffer Zone Control, Retrieval and Storage Strategies. This screen also shows, in real time, the current status of the S/R machine: Current Location, Current Operation Number, Operation Type, Bay Number and Co-Ordinates selected.

This screen is also used to control the semi-automatic movement of the ASRS. In semi-automatic mode the ASRS will travel to selected locations and perform chosen operations, i.e. store, retrieve, GoTo Dwell, Home and Reset.

ASRS layout and Contents

This screen, shown in Figure 4-4, displays the layout of the ASRS plus the contents of each bay. It is updated when the operator selects the 'Update' button. The information imported onto the screen at that point comes directly from the table 'ASRSContents' in the ASRS1.mdb database.

Class-Based Storage

This screen, shown in Figure 4-5, displays the classes allocated to each bay for class-based storage. It is updated when the operator selects the 'Update' button. The information imported onto the screen at that point comes directly from the table 'ASRSContents' in the ASRS1.mdb database.

Dedicated Storage

This screen, shown in Figure 4-6, displays the components allocated to each bay for dedicated storage. It is updated when the operator selects the 'Update' button. The information imported onto the screen at that point comes directly from the table 'ASRSContents' in the ASRS1.mdb database.

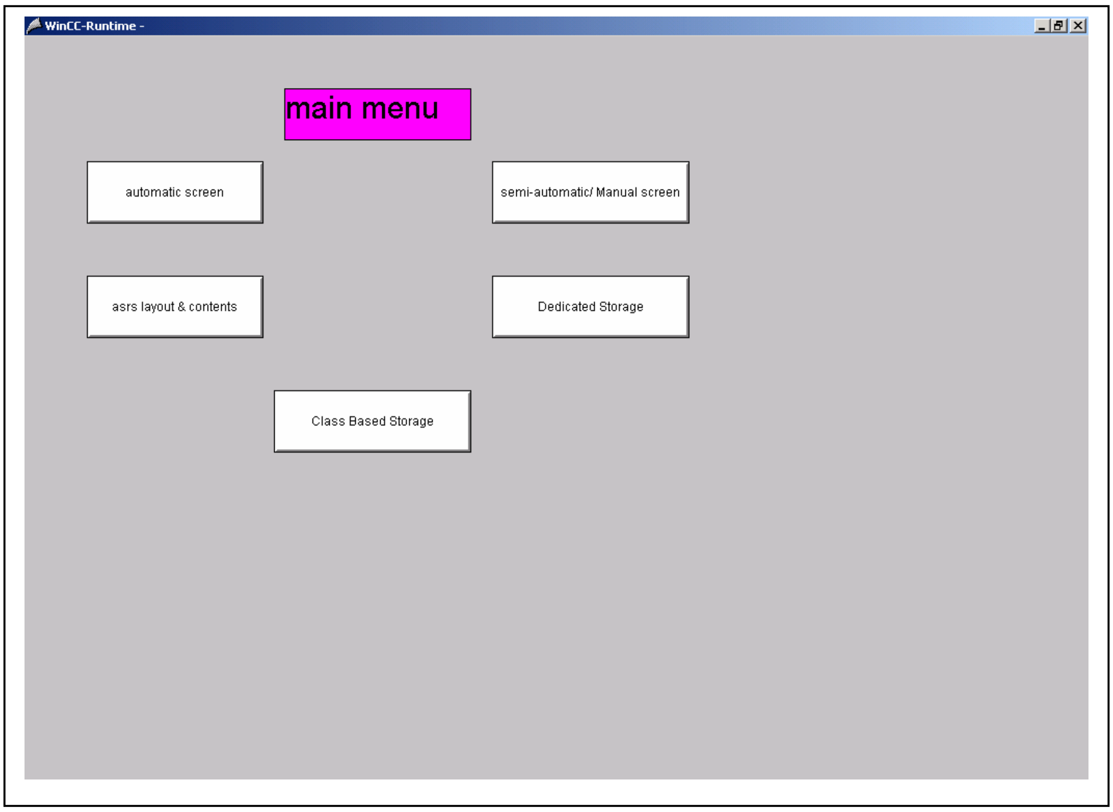


Figure 4-2: Main Menu

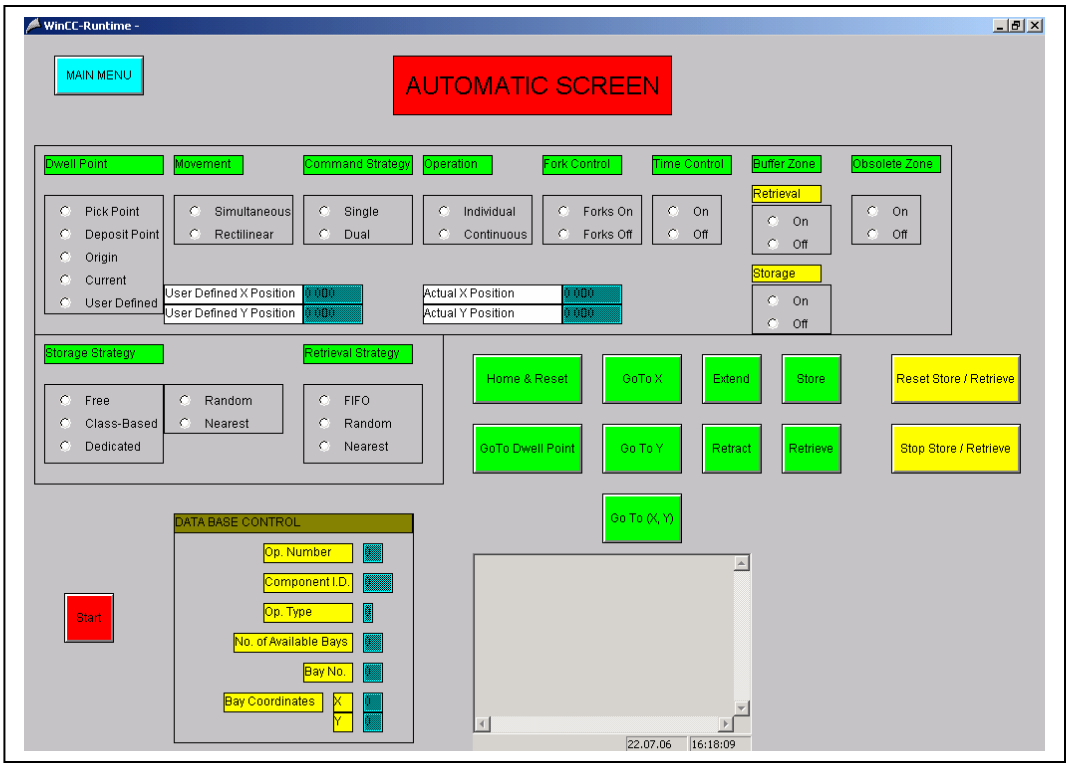


Figure 4-3: Semi / Full Automatic Screen

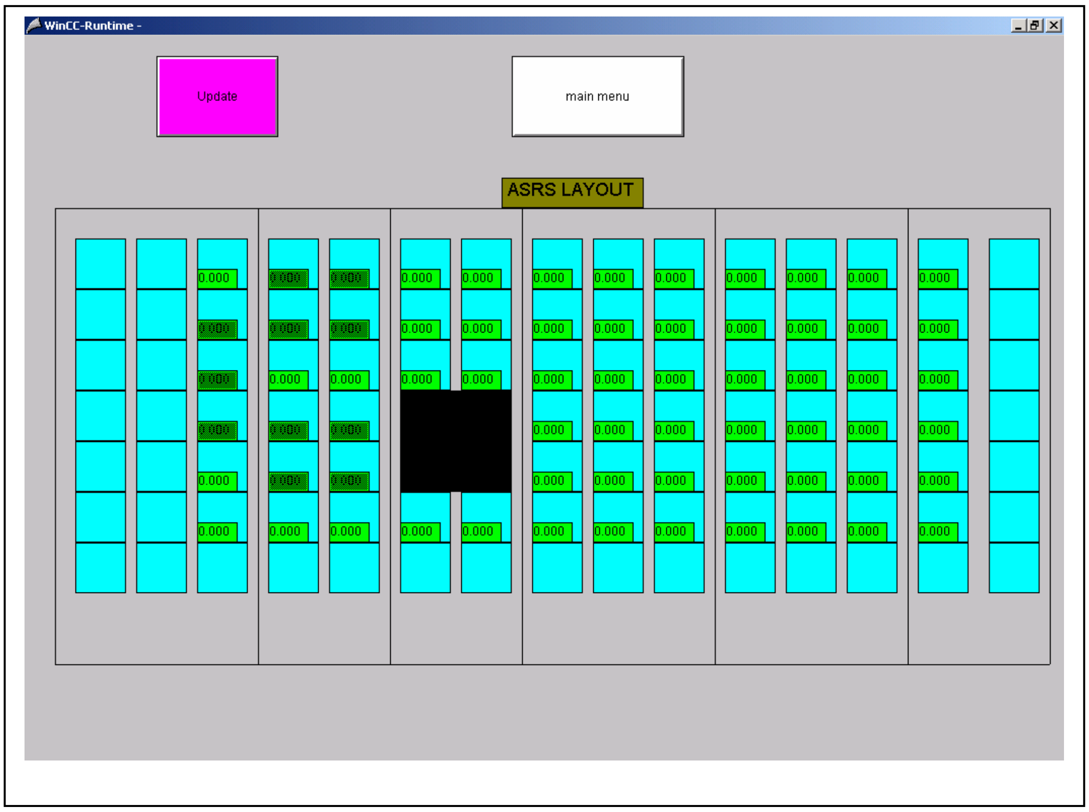


Figure 4-4: ASRS Layout and Contents

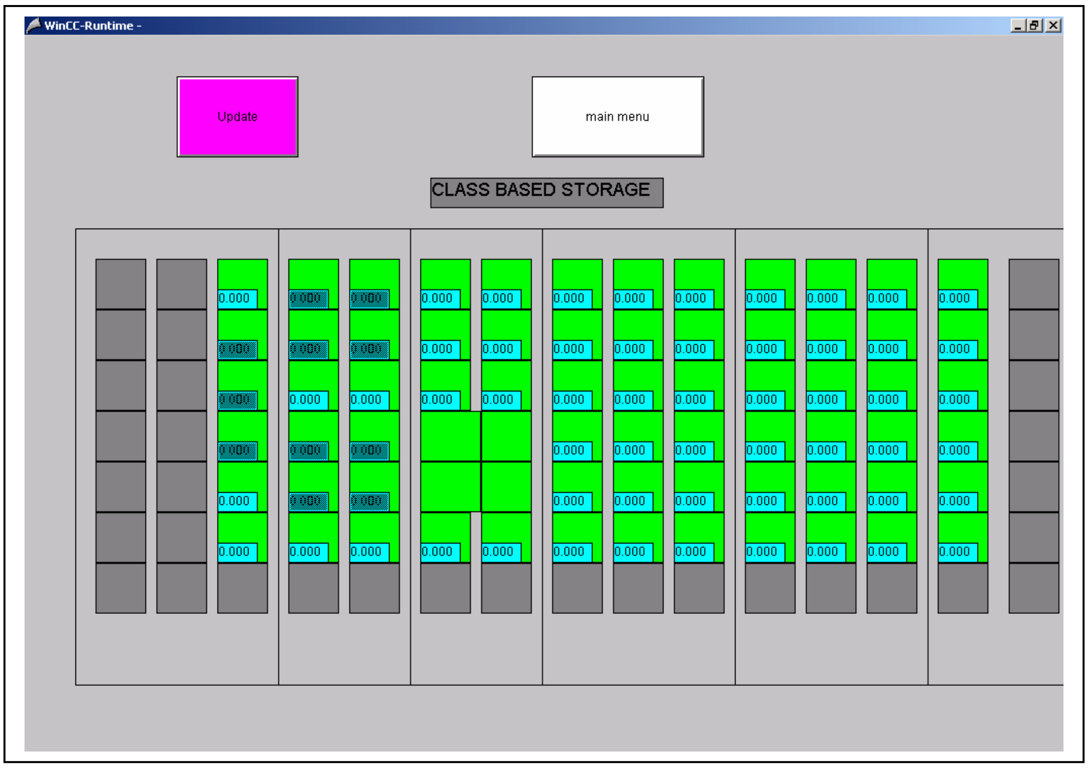


Figure 4-5: Class-Based Storage

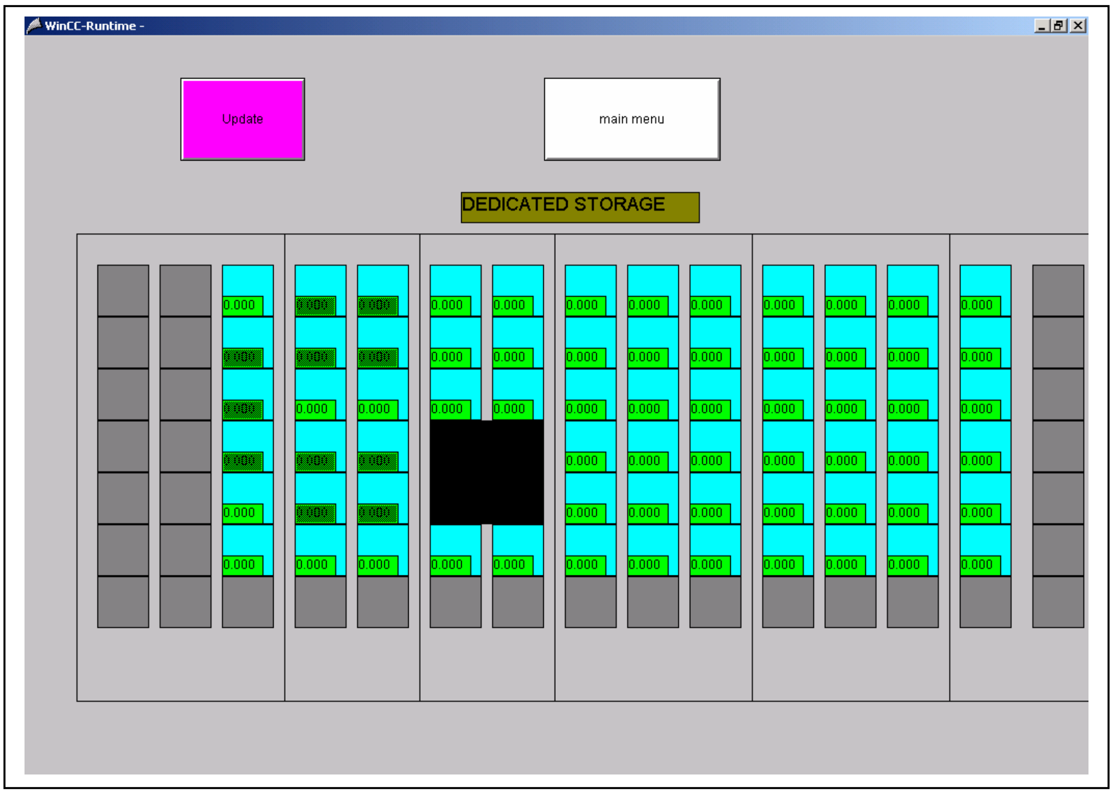


Figure 4-6: Dedicated Storage

4.3.3.2 Global Script – Project Functions

Project functions are C functions which are only valid in the actual project in which they were created. Project functions are used to make graphic objects and archives dynamic. They can also be used in other project functions and Global Script actions. There are five project functions developed for this research:

- ASRSOp
- ASRSOpAction
- SetPlaceCoOrds
- ASRSOpComp
- ASRSDwell

<u>ASRSOp</u>

This function pulls information from the Requirements table in the control database and determines the appropriate bay location for the next suitable item and then passes control onto either the project function 'ASRSOpAction' to update the appropriate co-ordinates for the S/R machine or else passes control to the global action 'GoToXYCompleteAction' if the S/R machine is already at the required position. This function is used as the initial starting point in the control sequence. Within this script the majority of the variables are defined. These are then re-defined as external variables in the other scripts in which they are required. Figure 4-7 shows the declaration of a structure variable for use with the WinCC function library DBExt.DLL and the MicroSoft Access database.

The structure variable shown in this figure is called 'CoOrdVar []', this array holds the BayNumber, XCoOrdinate, YCoOrdinate and ComponentID as variables: BayNo (long), XCo (int), YCo (int) and AutoID (long). This information is retrieved from the database table 'ASRSContents' in ASRS1.mdb using the command shown in Figure 4-8.

This is very similar to all the interactions with these structures whether they are retrieving data from database tables, putting data into tables, adding or deleting records in tables.

The section of script shown in Figure 4-9 is active when the 'Time Control' radio button is selected as "ON" in the 'Semi / Full Automatic' WinCC screen. This checks to see if the time difference between the required time and the actual time is greater than 5 seconds (arbitrarily chosen to minimise inaccuracy in completion of the

operation and to maximise the PC free time) and if so, then a tag bit is set high which calls a PLC function, FC 2, which returns control to this script after a 5 second delay. Thus ensuring that the requirements will be acted on near to the correct time.
Figure 4-10 shows a section of script that is replicated a number of times to select the appropriate sub-routine for the storage or retrieval strategy chosen in the 'Semi / Full Automatic' WinCC screen. The section shown sends control to the 'Random' sub-routine if the selection is Random and Free and the OpReqd, retrieved from the 'Requirements' table in the database, equals 1 for Store.
The sub-routines handling the storage and retrieval strategies are: 'Random', 'Nearest', 'Dedicated', 'Class-based' and 'FIFO'. These routines all work in a similar way:

- Retrieve the number of records corresponding to a search condition, i.e. 'Component ID = 101' from the 'ASRSContents' table.
- If no appropriate records exist then GoTo error1, where similar records are checked in the 'Requirements' table and then return to the start of the script to retrieve a different requirement for which records in the 'ASRSContents' table may exist
- If only one record exists then retrieve the Bay Number etc. from the 'ASRSContents' table and go to the 'Normal' Sub-routine which, as shown in Figure 4-11 sets various TagWords (external 16 bit unsigned values linked to logical connections and thus to the appropriate channels) and runs the script 'ASRSOpAction', which activates the PLC function FC 17 or FC 18 depending on the travel type selected in the 'Semi / Full Automatic' WinCC screen.
- If more than one appropriate record exists then, as Figure 4-12 shows, these records are loaded into a suitable array where calculations are performed to select a record based on the strategy chosen, i.e. random, nearest, class-based, dedicated or FIFO. The 'Normal' sub-routine is then activated which, as stated above, sets various TagWords and runs the script 'ASRSOpAction', which activates the PLC function FC 17 or FC 18 depending on the travel type selected in the 'Semi / Full Automatic' WinCC screen.

When a situation arises where there are no workable requirements available then a message is output to the operator stating "No Op. Available – Requirements Need to be Modified" and the entire control sequence ends. When the 'Requirements' table is modified this then triggers the sequence to begin again. In the future when a link is created between the ASRS control and the MS SQL Server database controlling the FMS then updating of the 'Requirements' table in this database will trigger the activation of this function.

This project function is called from three separate locations in the control sequence:

- Initially when the 'Start' button on the 'Semi / Full Automatic' WinCC screen is pressed.
- From 'ASRSOpComp' after the initial operation of a Dual Cycle.
- From 'ASRSDwell' for continuous operation

It hands on control by running 'ASRSOpAction' or by setting the tag 'ASRSOpContinuous' high which is required by FC 17 or FC 18 to complete before handing control over to the action 'GoToXYCompleteAction' or, when no suitable requirements are available, the programs ends.

When the S/R machine is already at the required bay then the TagBit 'AtXYPosition' is set high which activates the action script 'GoToXYCompleteAction'.

<u>ASRSOpAction</u>

This project function sets the destination co-ordinates for the S/R machine by forcing the TagWords 'xAUTOReqPos' and 'yAUTOReqPos' to either the co-ordinates of the pick point (4,3) or the co-ordinates of the chosen bay in 'ASRSOp'. It then sets the TagBit 'GoToXY' high, which activates the Step7 functions FC 17 or FC 18 which results in the S/R machine moving to the required position. It also includes code which determines if the S/R machine is already at the designated location, this being the case then the TagBit 'AtXYPosition' is set high which results in the script action 'GoToXYCompleteAction' being called. This portion of the script is shown in Figure 4-13.

This function is called from two separate locations:

- Directly from the project function 'ASRSOp' at the very beginning of the control sequence and after the initial operation in a dual operation.
- From the action 'GoToXYCompleteAction' when the control strategy is for continuous control and the S/R machine has just returned to the Dwell Point and is ready to begin a new command sequence.

SetPlaceCoOrds

This function script sets the destination for the S/R machine after it completes a pick operation. This destination will be either to the deposit point or to a chosen bay. It sets the TagWords 'xAUTOReqPos' and 'yAUTOReqPos' to the Deposit point (5,3) if the operation being conducted is a retrieve or to the chosen bay number if the operation is a store. See Figure 4-14 to view this code.

This function is called from the global action 'PickComplete'.

It returns control to either PLC function FC 17 or FC 18 by the setting high of the TagBit 'GoToXY'.

```
DBEXT_VARIABLE_STRUCT CoOrdVar [] = {
{"BayNumber",&BayNo,DBE_VS_TYP_VARIABLE},
{"XCoOrdinate",&XCo,DBE_VS_TYP_VARIABLE},
{"YCoOrdinate",&YCo,DBE_VS_TYP_VARIABLE},
{"ComponentID",&AutoID,DBE_VS_TYP_VARIABLE},
{NULL,NULL,0}
};
```

Figure 4-7: Declaration of Structure Variable in 'ASRSOp'

```
Sprintf(sqlCond,"%s", strSQLOpReqd);

Result =DBEGetVariables
(PictureName,MyDataSource,TableName2,sqlCond,Requirements,&MyError);
If (!Result)
{
printf("No Available Bays \n");
}
```

Figure 4-8: DBEGetVariables in 'ASRSOp'

```
if (GetTagBit("TimeControl") ==1)
{
//printf("DateTimeReqd = %lf\n", DateTimeReqd);
//printf("Atime = %lf\n", Atime);
TimeDifference = ((DateTimeReqd - Atime)*86400.0);

//printf("TimeDifference = %lf seconds \n", TimeDifference);
if (TimeDifference > 5)
{
SetTagBit("TimeDelay",1);
printf("5 Sec. Time Delay Operational \n");
goto end;

}

}// end if TimeControl
```

Figure 4-9: Time Control in 'ASRSOp'

```
if ((GetTagWord("Random") ==1) && (OpReqd ==1)&&(GetTagWord("Free")==1))
{
printf("Random Free Storage selected. \n");
goto random;

}//end if Random Free Storage
```

Figure 4-10: Sub-Routine Selection in 'ASRSOp'

```
normal:

//printf("%s\n", strMsg);
SetTagWord("Liam1",wRecs);
SetTagWord("Liam2",BayNo);
SetTagWord("autocomponentID",AutoID1);
SetTagWord("XCoOrdinateDB",XCo);
SetTagWord("YCoOrdinateDB",YCo);
SetTagWord("OpReqdOP", OpReqd);
SetTagWord("OpNumberOP", OpNumber);
if (GetTagBit("ASRSOpCompDual")==1)
{
ASRSOpAction();
}
if (GetTagBit("ASRSDwellContinuous")==1)
{
SetTagBit("PickReqd",1);
SetTagBit("ASRSDwellContinuous",0);
SetTagBit("ASRSOpContinuous",1);
if(GetTagBit("ASRSDwellAtXYPosition")==1)
{
SetTagBit("ASRSDwellAtXYPosition",0);
SetTagBit("AtXYPosition",1);
}
}
//printf("Going to GoToXY in ASRSOp \n");
if (GetTagBit("Start")==1)
{
ASRSOpAction();
}

OpNumber1=OpNumber;
OpNumber = 999;
NoResult = 0;
```

Figure 4-11: Normal Sub-Routine in 'ASRSOp'

```
switch (wRecs)
{
case 0 :
            strcpy (strMsg, "None Available");
if (OpReqd == 1)
      sprintf(sqlCond,"%s%d",strSQLOperationReqd,OpReqd);
if (OpReqd == 2)
      sprintf(sqlCond,"%s%dAND%s%d",strSQLOperationReqd,OpReqd,strSQLPartNumber,PartNo);

goto error1;

case 1 :

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
             break;
default :
strcat (sqlCond , " AND Checked = 0");
bCheck = 1;

for (r = 0; r < wRecs ; r++)
{

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
sprintf( sqlCond1, "%s%d", strSQLHoldBay, BayNo);
Result=DBESetVariables(PictureName,MyDataSource,TableName,sqlCond1,CheckedPut,&MyError);

BayArray[r] = BayNo;
}//end for r
strcpy(sqlCond, " Checked = 1 ");
bCheck = 0;
Result=DBESetVariables(PictureName,MyDataSource,TableName,sqlCond,CheckedPut,&MyError);
srand(time(NULL));
r=rand( )%wRecs; //random number  0 to No. of data items in the array
sprintf(sqlCond,"%s%d",strSQLHoldBay, BayArray[r]);

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
if   (!Result)
{
strcpy (strMsg, "Can't Get Co=ordinates");
goto error;
}
} // end switch wRecs
```

Figure 4-12: Loading of Appropriate Bays into Array in 'ASRSOp'

```
if (GetTagBit("Start")==1)
{
SetTagBit("Start",0);
SetTagBit("ASRSOpStart",1);

if((GetTagBit("DwellPickPoint")==1)&&(OpReqd == 1) &&
(GetTagWord("xAUTOActPos")==4) &&
(GetTagWord("yAUTOActPos")==3))
{
SetTagBit("AtXYPosition",1);
}//end if
else
if((GetTagBit("CurrentLocation")==1)&&(OpReqd == 2) &&
(GetTagWord("xAUTOActPos")==XCo) &&
(GetTagWord("yAUTOActPos")==YCo))
{
SetTagBit("AtXYPosition",1);
}//end if
else
if((GetTagBit("UserDefined")==1)&&(OpReqd == 2) &&
(GetTagWord("xAUTOActPos")==XCo) &&
(GetTagWord("yAUTOActPos")==YCo))
{
SetTagBit("AtXYPosition",1);
}//end if
else
{
SetTagBit("GoToXY",1);
}
}
```

Figure 4-13: Setting of 'AtXYPosition' in 'ASRSOpAction'

```
#include "apdefap.h"

extern int XPick, YPick, XDeposit, YDeposit, XCo, YCo;
extern long OpReqd;
void SetPlaceCoOrds()
{
printf("Start SetPlaceCoOrds \n");
if (OpReqd == 1)
{
SetTagWord("xAUTOReqPos",XCo);
SetTagWord("yAUTOReqPos",YCo);
}//end if
if (OpReqd == 2)
{
SetTagWord("xAUTOReqPos",XDeposit);
SetTagWord("yAUTOReqPos",YDeposit);
}//end if
//SetTagBit("PlaceReqd",1);
SetTagBit("SetPlaceCoOrdsPlaceReqd",1);
SetTagBit("GoToXY",1);
printf("End SetPlaceCoOrds \n");
}
```

Figure 4-14: 'SetPlaceCoOrds'

ASRSOpComp

This function updates the relevant database tables when an operation is completed and calls the project function 'ASRSOp' if dual command is operational otherwise it passes control to the PLC place function. This function, called from the action 'GoToXYCompleteAction' consists of a large number of externally declared variables and structure variables (see Figure 4-15). It updates the 'ASRSContents', 'Requirements' and 'Operations Completed' tables in the ASRS1.mdb database as the place operation is being completed.

It also creates two different control paths, one for initial Dual operations and another for Single or secondary Dual operations. The first path activates the 'ASRSOp' function to determine the bay co-ordinates for the secondary Dual operation whereas the second path does not require this calculation and immediately passes control to the PLC function FC 20.

When the command type chosen is 'Single' or when the initial operation in a 'Dual' command is completed then this function sets the TagBit 'ASRSOpCompSingle' high. This TagBit is required within the PLC function FC 20 to call the action 'PlaceComplete2' when both FC 20 and 'ASRSOpComp' are complete.

When the second operation in a Dual command is completed this function sets the TagBit 'ASRSOpCompDual' high and also calls the function 'ASRSOp'.

ASRSDwell

This function determines the co-ordinates for the dwell position and passes control to the appropriate PLC function to move the S/R machine to this location. For continuous operation this function passes control to the project function 'ASRSOp' to determine the location for the next requirement otherwise control is passed to the PLC dwell function and the S/R machine awaits operator intervention. This function is called from the action 'PlaceComplete2' which in turn is called from the PLC function FC 20 when the place operation of a 'Single' or secondary 'Dual' command is completed. This function sets the TagWords 'xAUTOReqPos' and 'yAUTOReqPos' to the co-ordinates of the selected dwell point, i.e. (4,3) for Pick Point, (5,3) for Deposit Point, (xActual, yActual) for Current Location, (0,0) for Origin and (xReqd, yReqd) for User Defined.

If the strategy selected is for continuous operation then this function sets the TagBit 'GoToXY' high which activates either FC 17 or FC 18, which brings the S/R machine

to the dwell point and it also calls the project function ‘ASRSOp’ which calculates the bay co-ordinates for the next requirement. It also sets the TagBit ‘ASRSDwellContinuous’ high which is used in the function ‘ASRSOp’.
If the strategy selected is non-continuous then this function sets the TagBit ‘ASRSDwellNotContinuous’ high which is used in the PLC function FC 17 or FC 18 to call the action ‘GoToXYCompleteAction’ and also sets the TagBit ‘GoToXY’ high which activates FC 17 or FC 18 bringing the S/R machine to the chosen dwell point. When the S/R machine is already at the dwell point then the TagBit ‘GoTo XY’ is set high which activates the action script ‘GoToXYCompleteAction’.
Study of the Flow Control diagram in Figure 4-1 shows that this is one of the functions in which control is split to ensure that calculations for the next required bay number are performed off-line thus ensuring no delays are encountered when the S/R machine arrives at the dwell point.

4.3.3.3 Global Actions
Global actions are used at run time to control the process and are executed by means of a trigger. There are four global actions developed for this project using WinCC script:

- GoToXYCompleteAction
- PickComplete
- PlaceComplete
- PlaceComplete2

<u>GoToXYCompleteAction</u>
This action is activated when the S/R machine reaches its destination passing control to a range of scripts or PLC functions, including, FC19 (Pick), FC 20 (Place), ‘ASRSOpComp’ and ‘ASRSOpAction’ depending on the status of a range of TagBits. This action is activated by the TagBit ‘AtXYPosition’ changing state. This tag is set high by the PLC functions FC17 and FC18 when the S/R machine reaches its required destination and also by the scripts ‘ASRSOp’, ‘ASRSOpAction’ and ‘ASRSDwell’ and the action ‘PlaceComplete’ if the S/R machine is already at the required position. All the actions are activated by the trigger tag changing state. In order to ensure that the code is only processed when the tag goes high this action contains the code ‘if

(GetTagBit("AtXYPosition")==1)' which tests for this tag going high. The other actions contain a similar test.

It is in this script that a test is performed as to whether 'ForksOn' has been selected on the 'Semi / Full Automatic' WinCC screen. This code, shown in Figure 4-16, will activate the required PLC function for automatic pick or place (FC 23 or FC 24).

The other functions performed by this action include:

- When TagBits 'PickReqd' and 'ASRSOpStart' are high (at pick point after initial selection of 'Start' on 'Semi / Full Automatic WinCC screen) then this action sets the TagBit 'RunPick' high which activates PLC function FC 19 or FC 23 resulting in an actual pick operation at the current location.
- When TagBits 'PickReqd' and 'PlaceCompletePickRequired' are high (at pick point after the initial place in a dual command) this action performs a pick at the current location as above.
- When TagBits 'PickReqd' and 'GoToXYCompleteActionPick' are high (at pick point at the start of a new requirement in continuous operation) this action performs a pick at the current location.
- When TagBits 'PlaceReqd' and 'SetPlaceCoOrdsPlaceReqd' are high (immediately after a pick operation) this action performs a place at the current location and also calls ASRSOpComp.
- When TagBits 'DwellComplete' and 'ASRSDwellNotContinuous' are high (S/R machine at dwell point) the control sequence ends waiting on the Start button to be re-pressed.
- When TagBits 'DwellComplete', 'ASRSOpContinuous' are high and TagBit 'ASRSOpNoRequirements' is low (at dwell point and about to move to new pick point in continuous operation) this action calls the project function 'ASRSOpAction' which sets the TagWords 'xAUTOReqPos' and 'yAUTOReqPos' to either the co-ordinates of the pick point (4,3) or the co-ordinates of the chosen bay depending on the operation type of the new requirement. This action also sets the TagBit 'GoToXY' high therefore activating the PLC function FC 17 or FC 18 moving the S/R machine to these new co-ordinates.
- When TagBits 'Dwell Complete', 'ASRSOpContinuous' and 'ASRSOpNoRequirements' are high (S/R machine at dwell point in

```
extern long BayNo, OpNumber, OpNumber1, OpReqd, PartNo, OpType, AutoID;
extern long BayNoRequired, CompID, AutoID1, ClassID; //CompID = DB Update, AutoID = DB
Search
extern long XAct, YAct, XStartPosn, YStartPosn;
extern long OpNo, DateTimeReqd, Checked;
extern long OpComp, DateTimeComp;
extern CMN_ERROR MyError;
extern char*  PictureName;
extern char MyDataSource[] ;
extern char sqlCond[100]  ;
extern char sqlCond1[100];
extern char sqlCond2[100];
extern char sqlCond3[100];
extern char strIOField[11] ;
extern char strSQLHoldComp[];
extern char strSQLDedicated[];
extern char strSQLHoldBay[];
extern char strSQLClass[];
extern char strSQLORDERBYDate[];
extern char strSQLOpReqd[];
extern char strSQLOpNumber[];
extern char strSQLNULL[];
extern char strSQLOperationReqd[];
extern char strSQLPartNumber[];
extern char TableName[];
extern char TableName1[];
extern char TableName2[];
extern char TableName3[];
extern char strFieldHold[];
extern char strMsg[20];
extern char strMsg1[20];
extern int  r, i, j, bCheck, XPick, YPick, XDeposit, YDeposit;
extern int XCo, YCo;
extern DWORD  wRecs;
extern struct tm *OldT;
extern time_t LiamTime;

extern DBEXT_VARIABLE_STRUCT CoOrdVarPut [];// = {
//{"ComponentID",&CompID,DBE_VS_TYP_VARIABLE},
//{"DateIn",&Atime,DBE_VS_TYP_VARIABLE},
//{NULL,NULL,0}
//};

extern DBEXT_VARIABLE_STRUCT CheckedPut [];// = {
//{"Checked",&bCheck,DBE_VS_TYP_VARIABLE},
//{NULL,NULL,0}
//};

extern DBEXT_VARIABLE_STRUCT OperationsCompleted [];// = {
//{"OperationNumber",&OpNumber,DBE_VS_TYP_VARIABLE},
//{"PartNumber",&PartNo,DBE_VS_TYP_VARIABLE},
//{"OperationComp",&OpComp,DBE_VS_TYP_VARIABLE},
//{"DateTime",&DateTimeComp,DBE_VS_TYP_VARIABLE},
//{NULL,NULL,0}
//};
```

Figure 4-15: 'ASRSOpComp'

```
if ((GetTagBit("PickReqd")==1)&&(GetTagBit("ASRSOpStart")==1))
{
if (GetTagBit("ForksOn")==1)
{
printf("ForksOn Selected \n");
SetTagBit("RunPick",1);
//printf("PickReqd is set to 1in GoToXYCompleteAction \n");
SetTagBit("PickReqd",0);
SetTagBit("ASRSOpStart",0);
SetTagBit("PlaceCompletePickReqd",0);

}// end if ForksOn
else
{
SetTagBit("RunPick",1);
printf("PictReqd is set to 1in GoToXYCompleteAction \n");
SetTagBit("PickReqd",0);
SetTagBit("ASRSOpStart",0);
SetTagBit("PlaceCompletePickReqd",0);

}// end else
}//end if PickReqd
```

Figure 4-16: 'GoToXYCompleteAction'

continuous operation but no suitable requirements available) the control sequence ends awaiting the updating of the 'Requirements' table.

PickComplete

This action is called after the completion of the PLC 'Pick' function (FC 19) and immediately calls the project function 'SetPlaceCoOrds' to change the destination for the S/R machine to the co-ordinates of the place location. This action is activated when the TagBit trigger 'PickComplete' changes state. Similar to the other actions it contains a test to ensure that the script is only ran when the trigger goes high.

PlaceComplete

This action is activated when the TagBit 'PlaceComplete' changes state. It is called by PLC function FC 20 after the completion of the place for the initial command in a dual command cycle. If there are no suitable requirements for the second dual command then this action calls the function 'ASRSDwell' to bring the S/R machine to its dwell point. If there are suitable requirements available then this action sets the TagBit 'GoToXY' high to activate the PLC function FC 17 or FC 18 to bring the S/R machine to the new pick location. If the S/R machine is already at this location then this action sets the TagBit 'AtXYPosition' high which activates the action 'GoToXYCompleteAction'.

PlaceComplete2

This action is activated when the TagBit 'PlaceComplete2' changes state. This action is called at the end of a place routine in a single or a secondary dual operation and calls the function 'ASRSDwell' to bring the S/R machine to its dwell point.

4.3.4 PLC Programming

The PLC program developed consists of a single organisation block (OB1) and thirteen functions (FC 1, 2, FC 14 – 24). The organisation block is called by the PLC operating system and is used to call the thirteen functions when required. OB1 allows the user to select a variety of strategies through conditional calling of the functions. This allows for completely isolated sections of code that are developed independently. The functions developed are listed in Table 3:

4.3.4.1 Manual Control X and Y Axes

This function is activated when an external switch, which is wired to I 2.4, is made. When this function is active the operation of a variety of externally wired switches allows for the manual control of both the X and Y axes. The over-travel sensors are still active in this function because all drives are hard-wired through these sensors, therefore protecting the S/R machine from over-zealous or inexperienced operators.

4.3.4.2 5 Second Time Delay

This function is activated when Time Control is selected on the 'Semi / Full Automatic' WinCC screen. When the difference between the current time and the time at which the action is required is greater than 5 seconds, control is then passed to this function from the project function 'ASRSOp'. Control passes back to the function 'ASRSOp' after the 5 seconds elapses.

Table 3: Actual PLC Blocks

Program item	**Description**
OB 1	Main Program Control
FC 1	Manual control, X and Y axes
FC 2	5 Second Time Delay
FC 14	Home & Reset
FC 15	GoTo X
FC 16	GoTo Y
FC 17	GoTo (X,Y) – Simultaneous Movement
FC 18	GoTo (X,Y) – Rectilinear Movement
FC 19	Pick – Time Delay
FC 20	Place – Time Delay
FC 21	Semi Automatic Pick
FC 22	Semi Automatic Place
FC 23	Automatic Pick
FC 24	Automatic Place

4.3.4.3 Home and Reset

Function 14 is activated during initial start-up or during a reset of the S/R machine. It is used to move the S/R machine to the extreme lower left hand corner of the racking

(0,0) and to reset the TagWords storing the actual S/R position, to zero. At this position there are two travel limit switches that are used by this function to indicate limits of travel in the horizontal and vertical directions. This function is activated from the Semi / Full Automatic screen in WinCC. When the S/R machine reaches the Home position the values for TagWords 'xAUTOActPos' (MW 745) and 'yAUTOActPos' (MW 725) are reset to zero.

4.3.4.4 Semi – Automatic Control (X and Y Axes)

Functions 15 and 16 allow for independent semi-automatic control of both the X and Y axes. The operator specifies the X and Y co-ordinates in the relevant input fields on the Semi / Full Automatic screen. The actual X and Y co-ordinates are displayed in the relevant output fields on the Automatic screen thus allowing the operator to do a real time check on the S/R machine's movement.

4.3.4.5 Full Automatic Control (X and Y Axes)

Functions FC 17 and FC 18 are used to move the S/R machine to the desired location specified by the TagWords 'xAUTOReqPos' (MW 755) and 'yAUTOReqPos' (MW 735). These functions are called from a number of different project and action scripts. FC 17 differs from FC 18 only in the way that the S/R machine travels. FC 17 is for simultaneous motion and FC 18 is for rectilinear motion (S/R machine moves to X position initially and then to the Y co-ordinate). Figure 4-17 shows the network responsible for the correct positioning of the S/R machine at the location required. Initially after 'Start' it is called from the project function, 'ASRSOp' to drive the S/R machine to the pick point (either (4,3) for a store operation or the chosen bay for a retrieve operation). Later on in the sequence of operations it is called from the function 'SetPlaceCoOrds' to drive the S/R machine to the place location (either (5,3) for a retrieve operation or to the required free bay for a store operation). FC 17 / FC 18 is called once more in the sequence from either 'PlaceComplete' or 'ASRSDwell' to move the S/R machine to either of two locations. In the second operation in a dual command the S/R machine has to be driven to a new pick point and in single operation the S/R machine is driven to the dwell point at which the operation will end (not continuous) or a new operation will commence (continuous).
FC 17 / FC 18 always results in the action script 'GoToXYCompleteAction' being called. This is called on the three occasions of the completion of FC 17 / FC 18 and depending on the status of various tags decisions are taken as to the path followed

within this action script. Figure 4-18 shows the network which results in the action script ‘GoToXYCompleteAction’ being called.

4.3.4.6 Pick / Place Time Delay

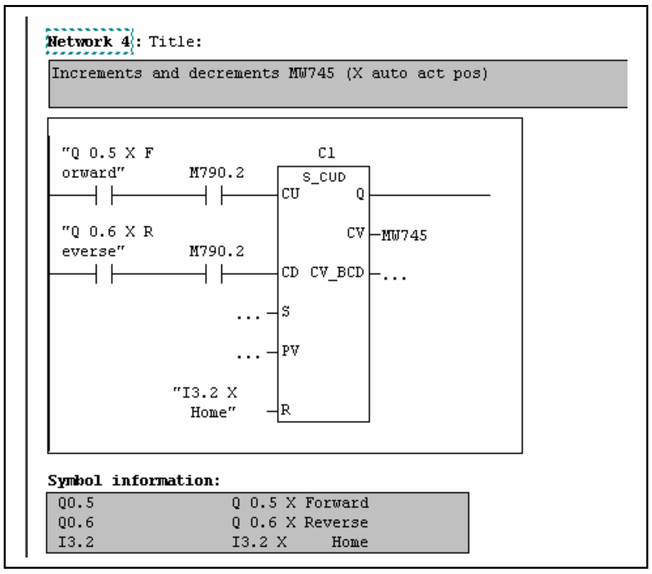
These functions (FC19 and FC 20) both use similar logic, see Figure 4-19, to create a five second delay imitating the actual time taken to perform either a pick or a place and are called from the global action ‘GoToXYCompleteAction’. They return control to quite different action scripts. A tag set high in FC 19 (see Figure 4-20) triggers the action script ‘PickComplete’. FC 20 returns control to either ‘PlaceComplete’ or ‘PlaceComplete2’ action scripts (see Figure 4-21) depending on whether the command type is dual or single.

4.3.4.7 Semi-Automatic Pick / Place

The two functions FC 21 and FC 22 are used for semi-automatic control of the forks. Control buttons on the ‘Semi / Full Automatic’ WinCC screen are used to set certain tags high which then activate either of these functions allowing the forks to perform a pick or a place at its current position. Figure 4-22 shows a portion of the code that controls the fork motion.

4.3.4.8 Automatic Pick / Place

Functions FC 23 and FC 24 are used to control the forks in automatic mode. They are called from the global action script ‘GoToXYCompleteAction’ when the relevant radio button on the ‘Semi / Full Automatic’ WinCC screen is selected. Figure 4-23 shows a portion of the ladder diagram used to control the pick action.

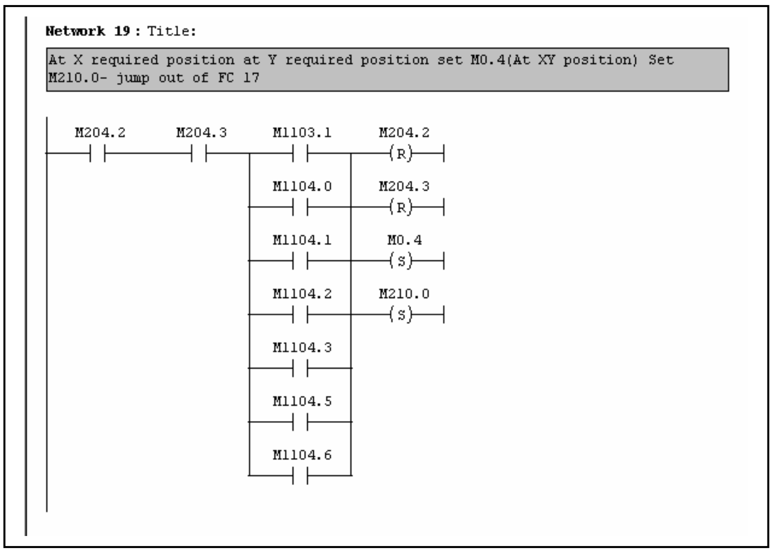

Network 4: Title:

Increments and decrements MW745 (X auto act pos)

Symbol information:

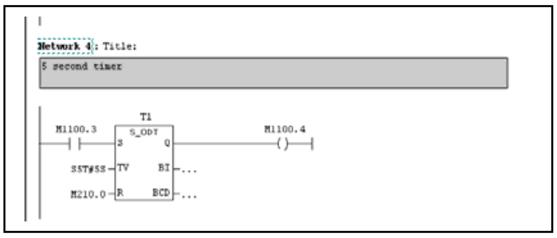
Q0.5	Q 0.5 X Forward
Q0.6	Q 0.6 X Reverse
I3.2	I3.2 X Home

Figure 4-17: X Axis Positioning in FC 17 / FC 18


Network 19 : Title:

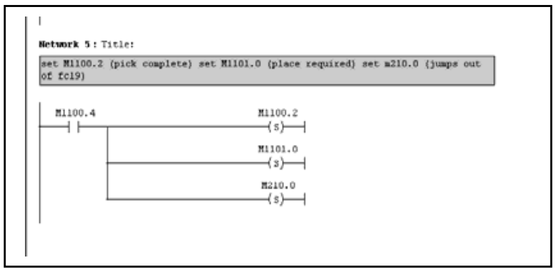
At X required position at Y required position set M0.4(At XY position) Set M210.0- jump out of FC 17

Figure 4-18: M 0.4 (AtXYPosition) being Set in FC 17 / FC 18


Network 4: Title:

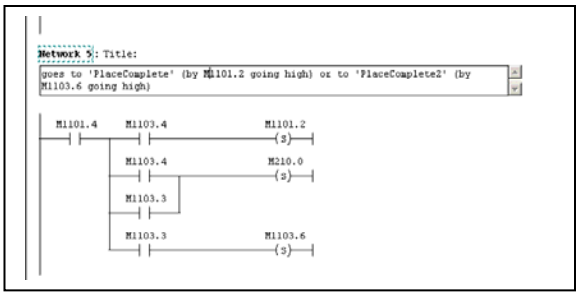
5 second timer

Figure 4-19: 5 Sec. Time Delay FC 19 / FC 20



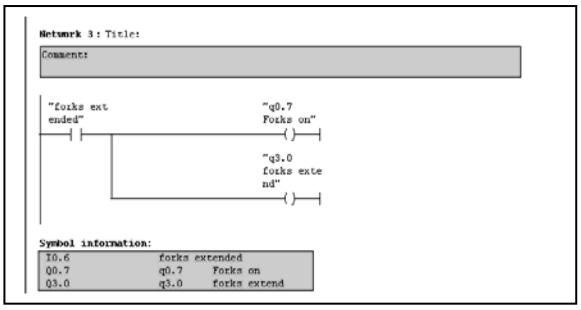
```
Network 5 : Title:
set M1100.2 (pick complete) set M1101.0 (place required) set m210.0 (jumps out
of fc19)
```

Figure 4-20: PickComplete being Set High in FC 19



```
Network 5 : Title:
goes to 'PlaceComplete' (by M1101.2 going high) or to 'PlaceComplete2' (by
M1103.6 going high)
```

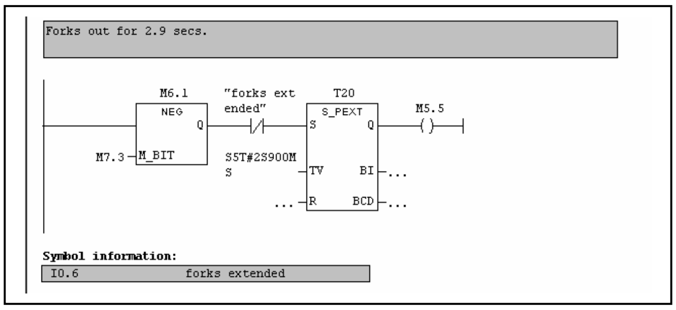
Figure 4-21: PlaceComplete or PlaceComplete2 being Called From FC 20



```
Network 3 : Title:
Comment:
```

```
Symbol information:
I0.6                forks extended
Q0.7                q0.7     Forks on
Q3.0                q3.0     forks extend
```

Figure 4-22: Forks Extend in FC 21



```
Forks out for 2.9 secs.
```

```
Symbol information:
I0.6                forks extended
```

Figure 4-23: Pick Control in FC 23

4.4 Communications

4.4.1 SCADA Communication with Database

A chart summarising the data flow between the WinCC SCADA program and the database is shown in Figure 4-24. With reference to this figure the communication process consists of:

1. Set Search Criteria

This stage sets the search criteria for choosing an appropriate job from the list of requirements.

Within the Project Function 'ASRSOp' the search criteria, sqlCond is initially set to 'OperationReqd = 1 OR OperationReqd = 2 AND Checked = 0'. In a dual cycle operation the search criteria will specify either a store (operation type 1) or a retrieve (operation type 2) for the second job search.

2. Retrieve Operation

This step retrieves the first job in the 'Requirements' table matching the search criteria set in 1. When no jobs exist of a particular type then the search criteria is changed to the alternative job type or else the entire process ends waiting on the 'Requirements' to be updated.

3. Determine Strategy

Within the Project Function 'ASRSOp' the storage or retrieval strategy selected by the operator on the 'Semi / Full Automatic' WinCC screen is used to pass control onto the appropriate program sub-routine. This sub-routine initially sets the search criteria for step 4.

4. Retrieve Count of Matching Records

The search criteria set in step 3 is used to determine the number of matching records in the database table 'ASRSContents'. This number is then tested using the Case command.

5. Flag Appropriate Records as Invalid

When zero records match the search criteria set in 4 then all appropriate demands are flagged, i.e. all store operations, all retrieve operations, all specific part number retrievals, all specific class type storage, all specific part number storage, etc.. Control then loops back to the initial search of the 'Requirements' table.

6. Retrieve Bay Co-Ordinates

When only one matching record exists in the table 'ASRSContents' then the co-ordinates of this bay are retrieved.

7. Retrieve All Matching Records

When more than one matching record exists then the 'Bay Number' of all matching records are retrieved into an Array.

8. Choose Record

A bay is chosen from all the matching records based on the storage or retrieval strategy selected by the operator. The co-ordinates of this bay are then retrieved.

9. Perform Operation

In this step various TagBits and TagWords are set 'High' passing control onto both PLC functions and other WinCC scripts to enable the performance of a store or retrieve operation. END is also set TRUE which terminates the search Do-While loop.

10. Do-While Loop

If no appropriate records are retrieved then control passes back to the initial search of the 'Requirements' table for a new operation.

11. End Loop

When an operation is performed or when no suitable operation is available then the loop completes and this script ('ASRSOp') finishes.

12. Operation Complete

When the store or retrieve operation is completed control passes to the project function 'ASRSOpComp' which updates a number of tables in the database.

13. Update Database

This step is used to complete the following:

Update 'ASRSContents' table

Un-flag records in 'Requirements' table

Delete record in 'Requirements' table

Add record to 'Operations Completed' table

4.4.2 Database

The MS Access database, ASRS1.mdb is accessible from the SIMUL8 Order Generator, the SIMUL8 ASRS Simulation and the WinCC Script through the ODBC data source 'ASRSDatabase'. This database consists of seven tables, three macros and eight queries (3 delete queries, 3 append queries and 2 update queries). These are all summarised below:

4.4.2.1 Database Tables

The seven tables are:

- ASRSContents
- ClassType
- InterTimes
- OperationsCompleted
- Requirements
- tmpRequirements
- tmpASRSContents

The structure for these tables is shown in Appendix D.

<u>ASRSContents</u>

This table (shown in Appendix D) stores the Co-Ordinates and contents of each bay plus the dedicated components and class type associated with these bays. The 'Checked' field is used by the WinCC project functions to highlight which bays have already been considered (i.e. as the array for free bays is being assembled for instance). The 'DateIn' field is used to store the date and time of the last movement into this record and is also used within the FIFO retrieval strategy in the script

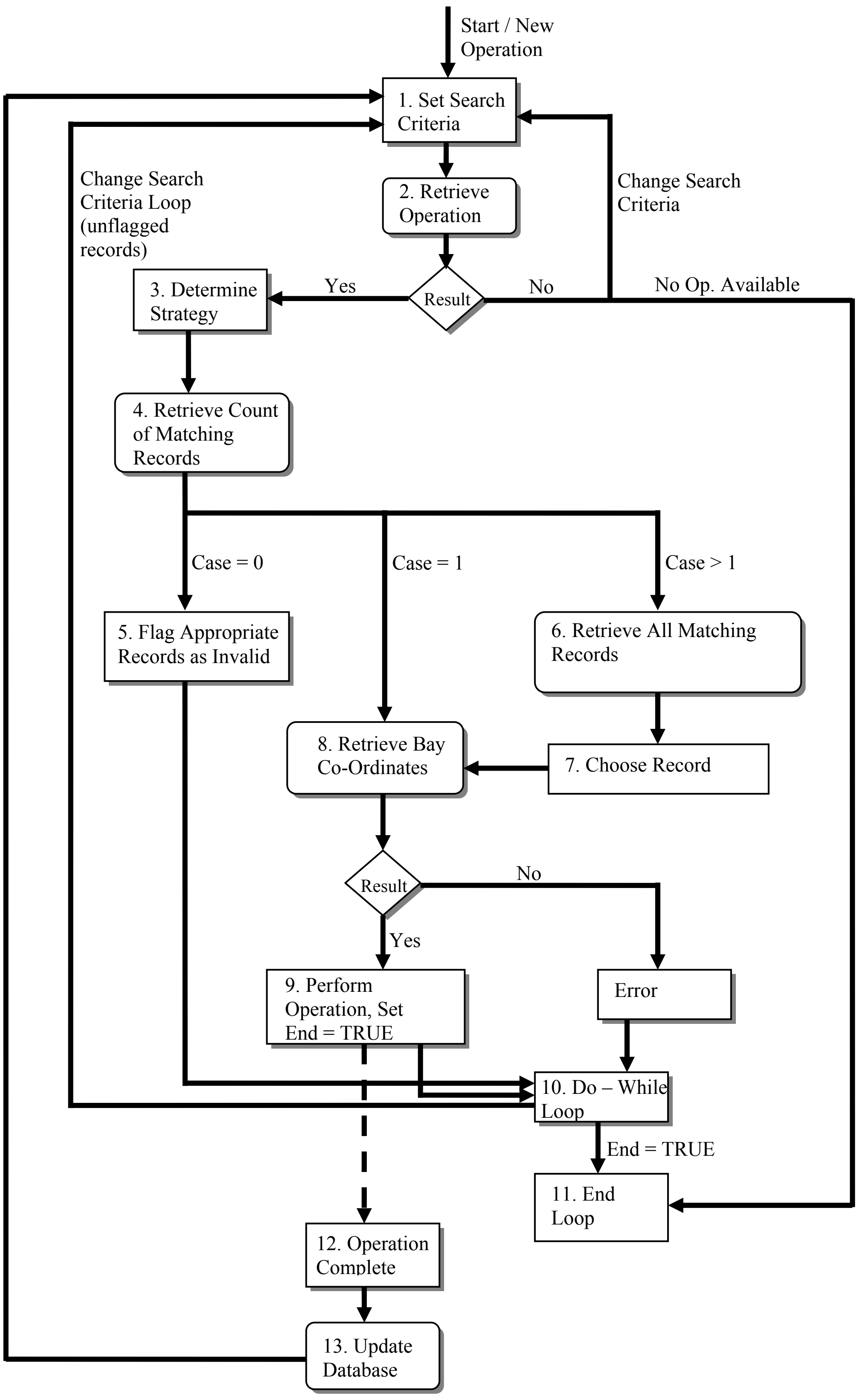


Figure 4-24: SCADA / Database Communication

‘ASRSOp’. The fields ‘KittingZone’ and ‘ObsoleteZone’ are check fields used to indicate which bays are to be used to store the obsolete or kitted components.

ClassType

This table (shown in Appendix D) allows the operator to associate the components with a class and is used by the project function ‘ASRSOp’ to select the appropriate bay for class-based storage.

InterTimes

This table (shown in Appendix D) is used to store the time intervals between requirements and is generated by the SIMUL8 Order Generator model. It is used by the query ‘qryUpdatetmpReqTimes’ which adds these time intervals to the ‘Start Time’ provided by the operator when prompted from this query as it updates the ‘Requirements’ table. This table is not relevant in standard operation mode when the orders are generated in real time by the FMS MRP system.

OperationsCompleted

This table (shown in Appendix D) is continuously updated by the project function ‘ASRSOpComp’ and contains information on the requirements completed to-date.

Requirements

This table (shown in Appendix D) is generated by the query ‘AddRequirements’ which moves the records from the table ‘tmpRequirements’ into this table. It is used by the project function ASRSOp when a new requirement is needed. The field ‘Checked’ in all the appropriate records is ticked when the ‘OperationReqd’ cannot be performed for the next available record. This field is unchecked by the project function ‘ASRSOpComp’ when a requirement is completed. This function also deletes the completed record.
In standard operation mode the requirements will be generated by an MRP system which will update this table. This action will trigger the relevant script to re-check the ‘Requirements’ table for any new additions which require immediate action.

tmpRequirements

The SIMUL8 Order Generator places the generated requirements into this table (shown in Appendix D) but does not enter the ‘DatetimeReqd’ details. This field is updated by the query ‘qryUpdatetmpReqTimes’ which adds the time intervals stored in the table ‘InterTimes’ by the Order Generator to the ‘Start Time’ provided by the operator when prompted by this query. This table is not relevant in standard operation mode when the orders are generated in real time by the FMS MRP system.

tmpASRSContents

This table (shown in Appendix D) is used by the query ‘qryUpdateASRSContents’ when the table ‘ASRSContents’ needs to be modified, i.e. to modify the contents of the ‘Dedicated’, ‘Class’, ‘ComponentID’, ‘KittingZone’ or ‘ObsoleteZone’ fields for a particular set of trials. This table is not relevant in standard operation mode when the orders are generated in real time by the FMS MRP system.

4.4.2.2 Database Queries

The eight queries are:

Delete Queries

- DeleteAllReqs
- DeleteASRSContents
- DeleteOperationsComp

Append Queries

- qryAddRequirements
- qryUpdateASRSContents
- qryUpdateReqs

Update Queries

- qryCompUpdateto0
- qryUpdatetmpReqTimes

DeleteAllReqs

This query deletes all records from the ‘Requirements’ table.

DeleteASRSContents

This query deletes all records from the 'ASRSContents' table.

DeleteOperationsComp

This query deletes all records from the 'OperationsCompleted' table.

qryAddRequirements

This query appends all records in the table 'tmpRequirements' to the table 'Requirements'.

qryUpdateASRSContents

This query appends all records in the table 'tmpASRSContents' to the table 'ASRSContents'.

qryUpdateReqs

This is an older version of the 'qryAddRequirements' query which appends all records from the 'tmpRequirements' table to the 'Requirements' table but ignores the 'Checked' field in the process.

qryCompUpdateto0

This Update query forces the field 'Component ID' in the table 'ASRSContents' to 0 and the field 'DateIn' in the same table to the current date and time using the formula Now().

qryUpdatetmpReqTimes

This query consists of the following code:

```
UPDATE tmpRequirements INNER JOIN InterTimes ON
[tmpRequirements].[OpNumber]=[InterTimes].[OpNumber] SET
tmpRequirements.DateTimeReqd =
Now()+[InterTimes]![InterOpTime]+StartTime/60/24
WHERE (([tmpRequirements]![OpNumber]=[InterTimes]![OpNumber]));
```

It is used to update the 'DateTimeReqd' field in the 'tmpRequirements' table by combining the 'InterOpTime' field in the 'InterTimes' table with a variable called 'StartTime' (converted into a fraction of a day) and the current date and time (Now()).

4.4.2.3 Database Macros

The three macros are:

- Start Procedure
- mcrUpdateRequirements
- mcrUpdateREQSandASRSContents

<u>Start Procedure</u>

This macro deletes all existing records in the 'Requirements' and 'OperationsComp' tables, appends all records from the 'tmpRequirements' table to the 'Requirements' table and updates the 'ComponentID' field in the 'ASRSContents' table to 0. It consists of the following queries:

DeleteAllReqs
qryUpdateReqs
DeleteOperationsComp
qryCompUpdateto0

<u>mcrUpdateRequirements</u>

This macro deletes all the records in the 'Requirements' and 'OperationsComp' tables , updates the 'tmpRequirements' table with the calculation of the 'DateTimeReqd' and appends these records to the 'Requirements' table and also updates the 'ComponentID' field to 0 in all the records of the 'ASRSContents' table. It consists of the following queries:

DeleteAllReqs
qryUpdatetmpReqTimes
queryAddRequirements
DeleteOperationsComp
qryCompUpDateto0

<u>mcrUpdateREQSandASRSContents</u>

This macro deletes all the records in the 'ASRSContents', 'Requirements' and 'OperationsComp' tables, updates the 'tmpRequirements' table with the calculation of the 'DateTimeReqd' and appends these records to the 'Requirements' table and also appends the records in the 'tmpASRSContents' table to the 'ASRSContents' table. It consists of the following queries:

DeleteASRSContents

qryUpdateASRSContents

DeleteAllReqs

qryUpdatetmpReqTimes

queryAddRequirements

DeleteOperationsComp

4.4.3 Order Generator

This program allows for the generation of requirements for the ASRS. Using the menu system shown in Figure 4-25 the operator can alter the following:

Product Range

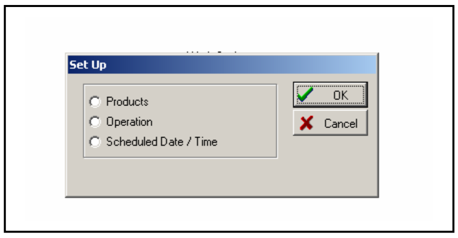
Operation Type

Scheduled Date / Time

to generate an unlimited quantity of requirements which are transferred into the tables 'InterTimes' and 'tmpRequirements' of the MSAccess database, ASRS1.mdb, through the ODBC link 'ASRSDatabase'. Figure 4-26 shows the spreadsheet 'VarProducts' which holds the Product Range information. The spreadsheets for the Operation Type and Scheduled Date / Time are shown in Figure 4-27 and Figure 4-28 below.

The main screen of this model is shown in Figure 4-29, when the operator 'Runs' this schedule a prompt appears as shown in Figure 4-30, asking for the number of operation to generate. The generated orders are then displayed as 'spSchedule', see Figure 4-31, and the operator is requested if the database schedule is to be updated, Figure 4-32.

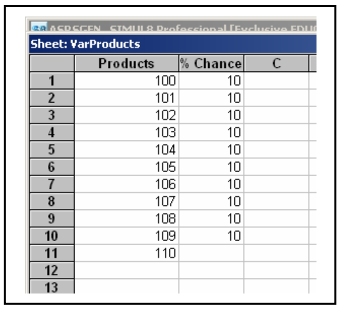
The benefit of splitting the generated schedule in two and updating the two tables is that this allows the operator to re-run trials using this schedule and the same date/time requirement even though the new trials may be running hours or days later than the original trial.



Set Up

- Products
- Operation
- Scheduled Date / Time

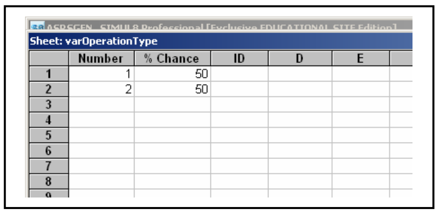
Figure 4-25: Menu System



Sheet: VarProducts

	Products	% Chance	C
1	100	10	
2	101	10	
3	102	10	
4	103	10	
5	104	10	
6	105	10	
7	106	10	
8	107	10	
9	108	10	
10	109	10	
11	110		
12			
13			

Figure 4-26: ssVarProducts

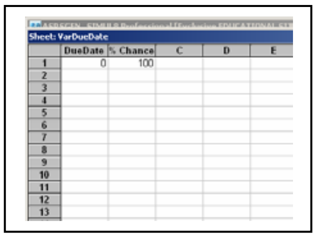


ASPSGEN - SIMUL8 Professional [Exclusive EDUCATIONAL SITE Edition]

Sheet: varOperationType

	Number	% Chance	ID	D	E
1	1	50			
2	2	50			
3					
4					
5					
6					
7					
8					

Figure 4-27: ssVarOperationType



Sheet: VarDueDate

	DueDate	% Chance	C	D	E
1	0	100			
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

Figure 4-28: ssVarDueDate

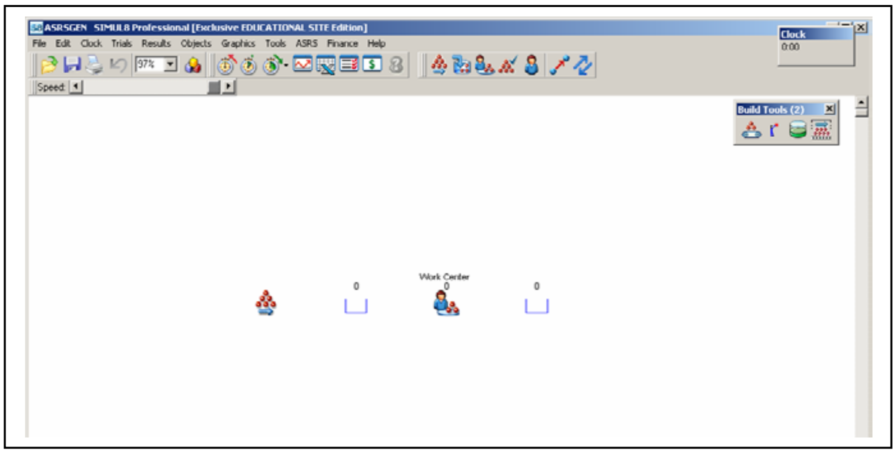
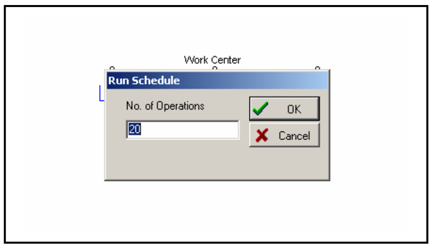

ASRSGEN SIMUL8 Professional [Exclusive EDUCATIONAL SITE Edition]
File Edit Clock Trials Results Objects Graphics Tools ASRS Finance Help
97%
Speed
Clock
0:00
Build Tools (2)
Work Center
0
0
0

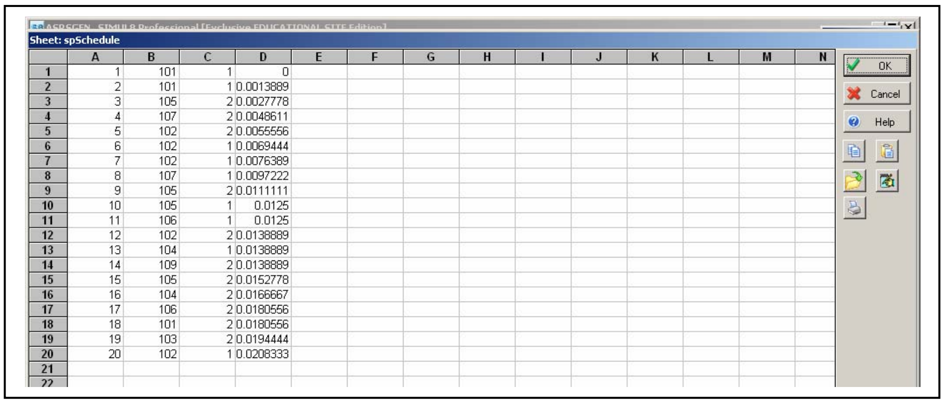
Figure 4-29: Order Generator Main Screen



Run Schedule

No. of Operations: 20

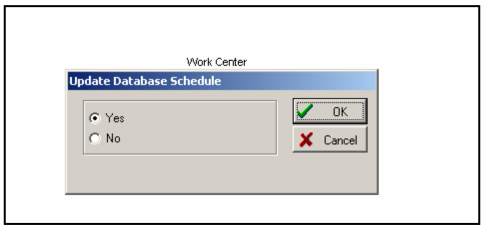
Figure 4-30: Operator Prompt re Order Number


ASRSGEN SIMUL8 Professional [Exclusive EDUCATIONAL SITE Edition]
OK
Cancel
Help

Sheet: spSchedule

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	1	101	1	0										
2	2	101	1	0.0013889										
3	3	105	2	0.0027778										
4	4	107	2	0.0048611										
5	5	102	2	0.0055556										
6	6	102	1	0.0069444										
7	7	102	1	0.0076389										
8	8	107	1	0.0097222										
9	9	105	2	0.0111111										
10	10	105	1	0.0125										
11	11	106	1	0.0125										
12	12	102	2	0.0138889										
13	13	104	1	0.0138889										
14	14	109	2	0.0138889										
15	15	105	2	0.0152778										
16	16	104	2	0.0166667										
17	17	106	2	0.0180556										
18	18	101	2	0.0180556										
19	19	103	2	0.0194444										
20	20	102	1	0.0208333										
21														
22														

Figure 4-31: Generated Orders in spSchedule



Update Database Schedule

[✓ Yes] No

Figure 4-32: Operator Prompt for Database Update

4.5 Summary

The chapter summarises the development and control of the ASRS test-bed concentrating on the Step7, WinCC and MSAccess interaction. A short description of the SIMUL8 order generator model is also given at the end of this chapter.
The next chapter describes the mathematical model and summarises the results obtained from it and compares them with those obtained from the physical model.

5 ASRS Modelling and Testing

5.1 Introduction

This chapter concentrates on the modelling and analyses of the proposed ASRS and predicts its performance under a variety of work conditions and control strategies. The simulation model, developed by Walsh (2004), used to analyse this design has the flexibility to include all the various strategies discussed in Chapter 3.

This model was developed using SIMUL8 which is an object oriented software application. This software also has a proprietary programming language (Visual Logic®) incorporating a relatively straightforward editor. The proposed ASRS in W.I.T. is a single tier rack with 6 levels and 12 columns. The pick and deposit points utilise 4 bays, thus reducing the number of bays available for storage to 68. The storage bays, designed to handle a standard pallet, are all of equal dimensions but are not uniformly spaced, either horizontally or vertically. The pick-up and deposit conveyors are incorporated into the rack on level 3, columns 4 and 5. The horizontal and vertical drives can be driven individually or simultaneously and at fixed speeds. Three work centres are used to model the S/R machine, work centre 1 is for pre-loaded movement between the dwell point and the pick point, work centre 2 is for loaded movement between the pick point and the deposit point and work centre 3 is for post-loaded movement between the deposit point and the dwell point.

This chapter discusses the options available in the SIMUL8 model and summarises the results from a number of trials designed to test the capabilities of the proposed ASRS models with conclusions drawn on these trials.

5.2 Mathematical Model

5.2.1 User Interface

The simulation application contains the essential model components for the S/R machine, load pick and deposit points, load input generator and load demand generator. The model objects representing the storage bays are created when the user defines the capacity and layout of the single-tier storage rack (see main screen, Figure 5-1). The application is menu driven (see Figure 5-2) and requires the following information to be entered prior to simulation.

Create rack: User enters the ordinates of the rack levels (rows) and columns in a spreadsheet and the storage bay objects are created in the model. Any particular model configuration can be saved as a file and repeatedly re-used.

Operational Parameters:

- Shuttle horizontal and vertical travel speeds
- Shuttle Travel – Rectilinear or simultaneous horizontal and vertical
- P & D load / unload times
- Storage bay put and retrieve times
- Pick point location
- Deposit point location

Control Strategies:

- Storage – Random, Nearest, Dedicated or Class-based. In the case of dedicated and class-based the user is required to specify the bay allocations on a spreadsheet (see Figure 5-3)
- Retrieval – FIFO or Random (see Figure 5-4)
- Command – Single or Dual
- Dwell point – Pick Point, Deposit Point, Current Location or User defined.

Input / Output:

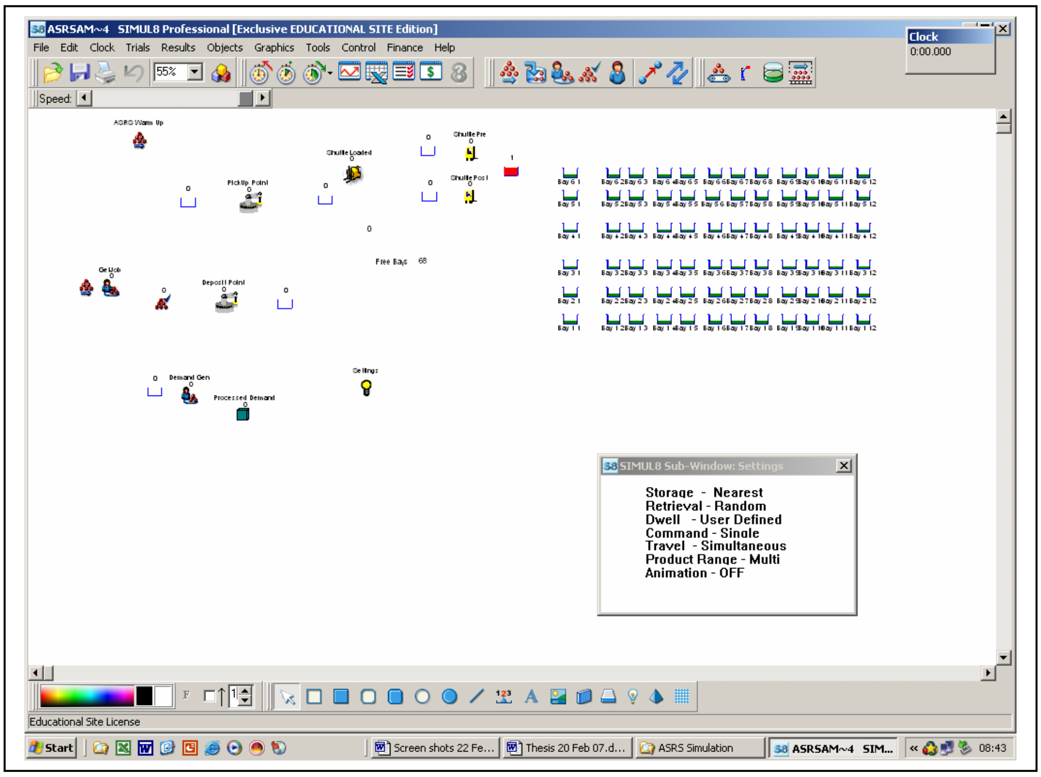
- ASRS input frequency
- Input load item (product) type
- ASRS demand frequency
- Demand item type

Simulation Conditions:

- Warm-up period
- Results collection period

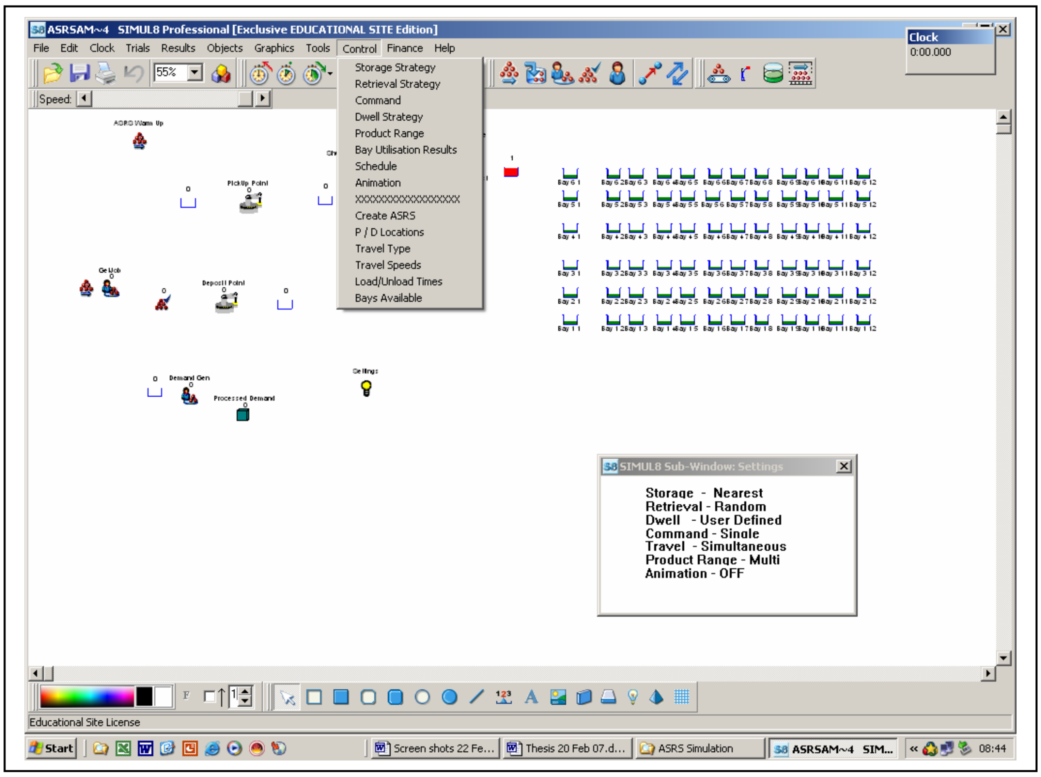
Results Options:

- View or output to an Excel spreadsheet (See Figure 5-5).



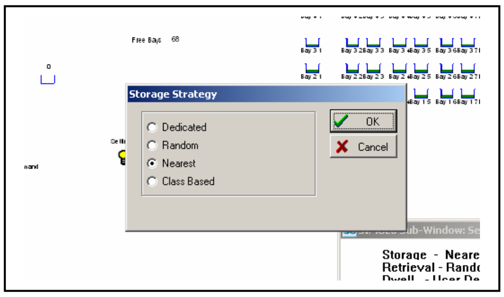
Storage - Nearest
Retrieval - Random
Dwell - User Defined
Command - Single
Travel - Simultaneous
Product Range - Multi
Animation - OFF

Figure 5-1: ASRS Simulator – Main Screen



Storage - Nearest
Retrieval - Random
Dwell - User Defined
Command - Single
Travel - Simultaneous
Product Range - Multi
Animation - OFF

Figure 5-2: ASRS Simulator – Control Menu

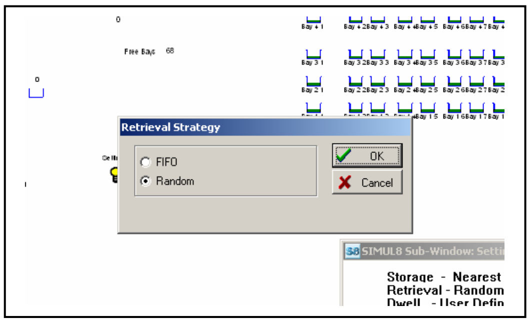


Storage Strategy

( ) Dedicated
( ) Random
[✓ Nearest]
( ) Class Based

Storage - Neare
Retrieval - Rand

Figure 5-3: Storage Strategy

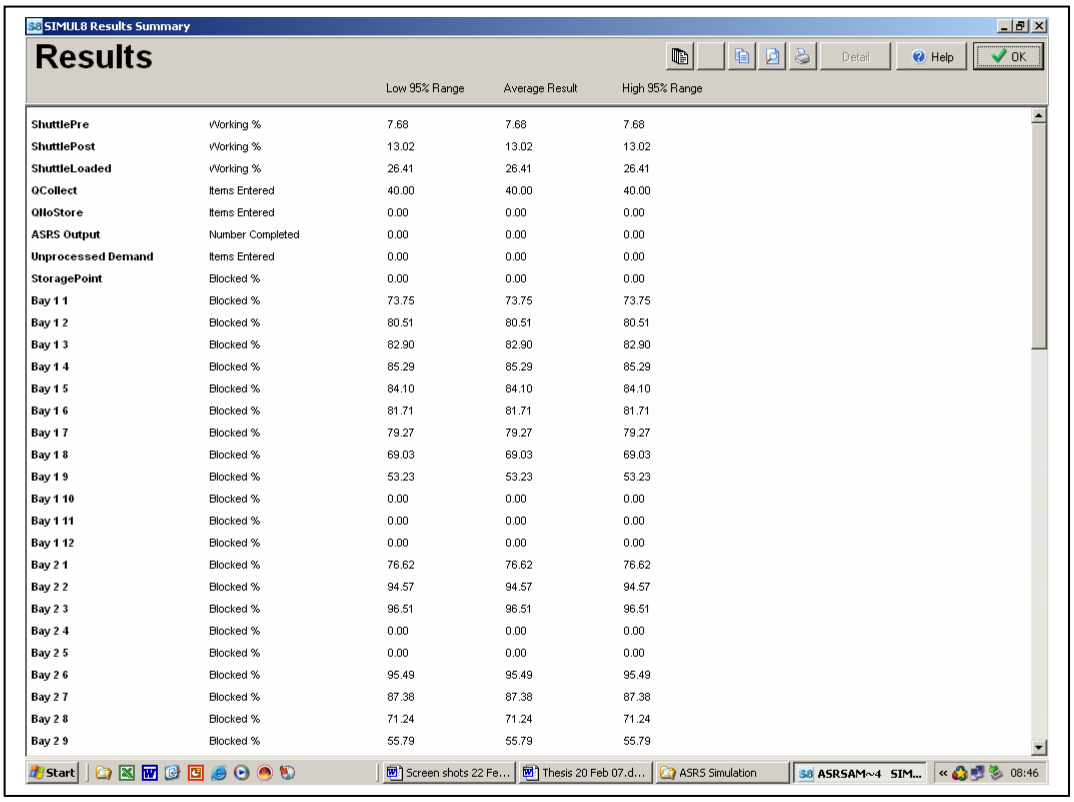


Retrieval Strategy

( ) FIFO
[✓ Random]

Storage - Nearest
Retrieval - Random
Dwell - User Defin

Figure 5-4: Retrieval Strategy



Results

		Low 95% Range	Average Result	High 95% Range
ShuttlePre	Working %	7.68	7.68	7.68
ShuttlePost	Working %	13.02	13.02	13.02
ShuttleLoaded	Working %	26.41	26.41	26.41
QCollect	Items Entered	40.00	40.00	40.00
QNoStore	Items Entered	0.00	0.00	0.00
ASRS Output	Number Completed	0.00	0.00	0.00
Unprocessed Demand	Items Entered	0.00	0.00	0.00
StoragePoint	Blocked %	0.00	0.00	0.00
Bay 1 1	Blocked %	73.75	73.75	73.75
Bay 1 2	Blocked %	80.51	80.51	80.51
Bay 1 3	Blocked %	82.90	82.90	82.90
Bay 1 4	Blocked %	85.29	85.29	85.29
Bay 1 5	Blocked %	84.10	84.10	84.10
Bay 1 6	Blocked %	81.71	81.71	81.71
Bay 1 7	Blocked %	79.27	79.27	79.27
Bay 1 8	Blocked %	69.03	69.03	69.03
Bay 1 9	Blocked %	53.23	53.23	53.23
Bay 1 10	Blocked %	0.00	0.00	0.00
Bay 1 11	Blocked %	0.00	0.00	0.00
Bay 1 12	Blocked %	0.00	0.00	0.00
Bay 2 1	Blocked %	76.62	76.62	76.62
Bay 2 2	Blocked %	94.57	94.57	94.57
Bay 2 3	Blocked %	96.51	96.51	96.51
Bay 2 4	Blocked %	0.00	0.00	0.00
Bay 2 5	Blocked %	0.00	0.00	0.00
Bay 2 6	Blocked %	95.49	95.49	95.49
Bay 2 7	Blocked %	87.38	87.38	87.38
Bay 2 8	Blocked %	71.24	71.24	71.24
Bay 2 9	Blocked %	55.79	55.79	55.79

Figure 5-5: Results Summary

5.2.2 Storage Bays

The storage bays, although modelled as work centre objects, are defined by the ordinates of the rows and columns, which are specified by the user when building the rack. The data is stored in an internal spreadsheet. There is a facility to denote any rack locations not available for storage. This feature allows for P & D points to be located at any position in the rack. The status of all bays is recorded in two other spreadsheets; the stored product ID or blank if the bay is available and the time of storage of a load. A further spreadsheet is employed to specify dedicated storage.
Each bay is modelled as a work centre object with zero work time and no output route. When a retrieval is required from a particular bay an output route is created, the work object released and the output route removed. A work centre object collects data on the total time that a work centre is blocked which in this case is the total time that a bay is in use.

5.2.3 ASRS Input

The Work Entry object (ASRS Input) determines the unit loads (Work Item object) inputted to the ASRS model. The user sets the probability distribution to control inter-arrival times. Only one Work Item type is used and different products are distinguished by a Label value attached to the item. The ASRS input object assigns the product ID values, again using a probability distribution set by the user.
All items entering the system are passed to the queue for the Pick Point work centre object. The exit event logic performs the tasks of calculating the shuttle travel time to the pick-up location, determining the storage bay and then calculating the travel time to the designated bay, including load and unload times. The Pick Point also carries out the tasks of linking the shuttle to the storage bay objects. In the case that there is not an available bay for the particular load, the item is routed to the QNoStore queue and recycled to the front of the queue for pick-up by incrementing a recycle tag. This model also allows for the inputting of requirements (both Storage and Retrieval) from a Requirements table in the ASRS Database which is used to control the physical ASRS.

5.2.4 The S/R Machine

The S/R machine is modelled using three work centre objects which simulate the three stages of the shuttle operation; travel empty to collect a load (pick-up point or storage bay – ShuttlePre); travel loaded to deliver the load (storage bay or deposit point –

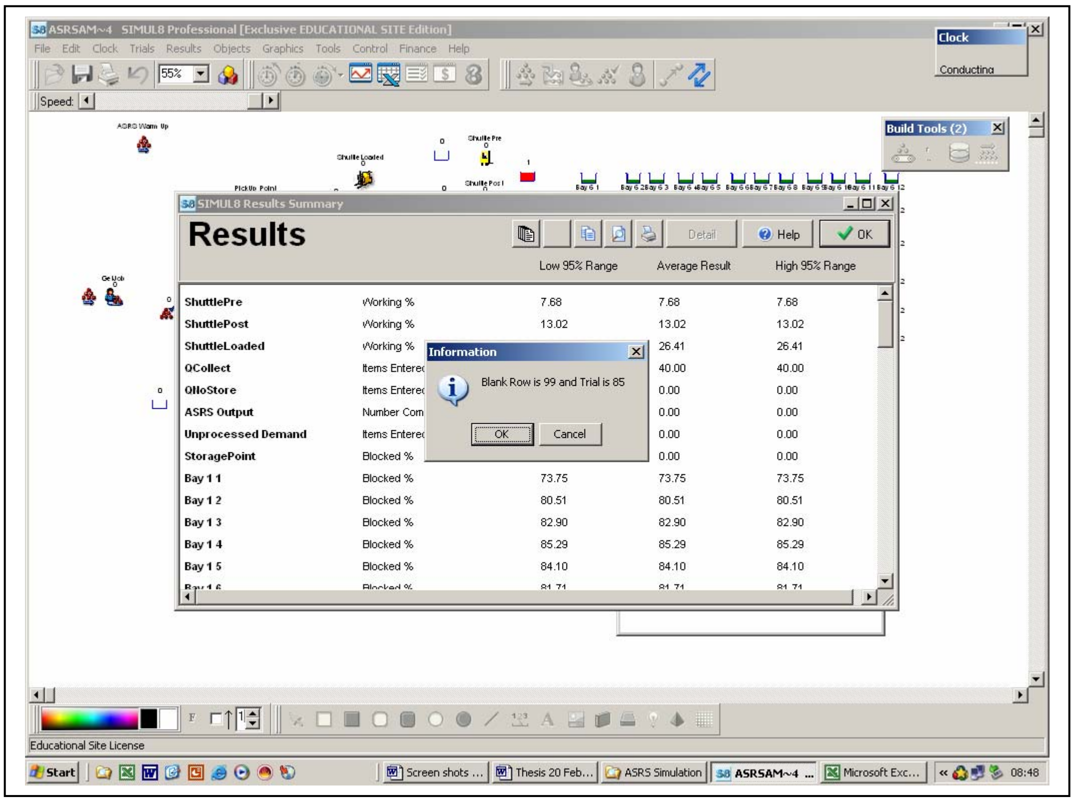
ShuttleLoaded); travel empty to dwell point (ShuttlePost). Enabling the events, of the start and end of each motion, gives greater control of the model, particularly under dual command conditions.

5.2.5 ASRS Demand

The work entry object (ASRS Demand) creates the demand product ID and inter-arrival times. The Demand Gen work centre determines the bay location, links the appropriate bay, the S/R machine and the deposit point object. Where the item required is not available the demand item is routed to the Unprocessed Demand queue. The reasoning here is similar to that for the handling of unstored loads. The work entry point is not linked to the queue for the Demand Gen. This link is automatically enabled at the end of the warm-up period, thus allowing the ASRS to be partially (or fully) loaded at the start of a run.

5.2.6 Results Display

On completion of a run, collected and calculated results data are presented in spreadsheet format, and if required by the user output to MS Excel (see Figure 5-6). This output includes the calculated average cycle times (dual and single command times where appropriate), operating times for each of the three stages of the S/R machine, the number of bays full at the start and end of the run, loads stored (and loads rejected) and loads retrieved (and unavailable loads). Additional data such as individual bay utilisation and product type of rejected loads is available by accessing the model objects and the standard results summary. Where the user initiates a trial, consisting of a number of runs, the data output will be the averages over the number of runs and also includes the ± 99% range (see Figure 5-7).

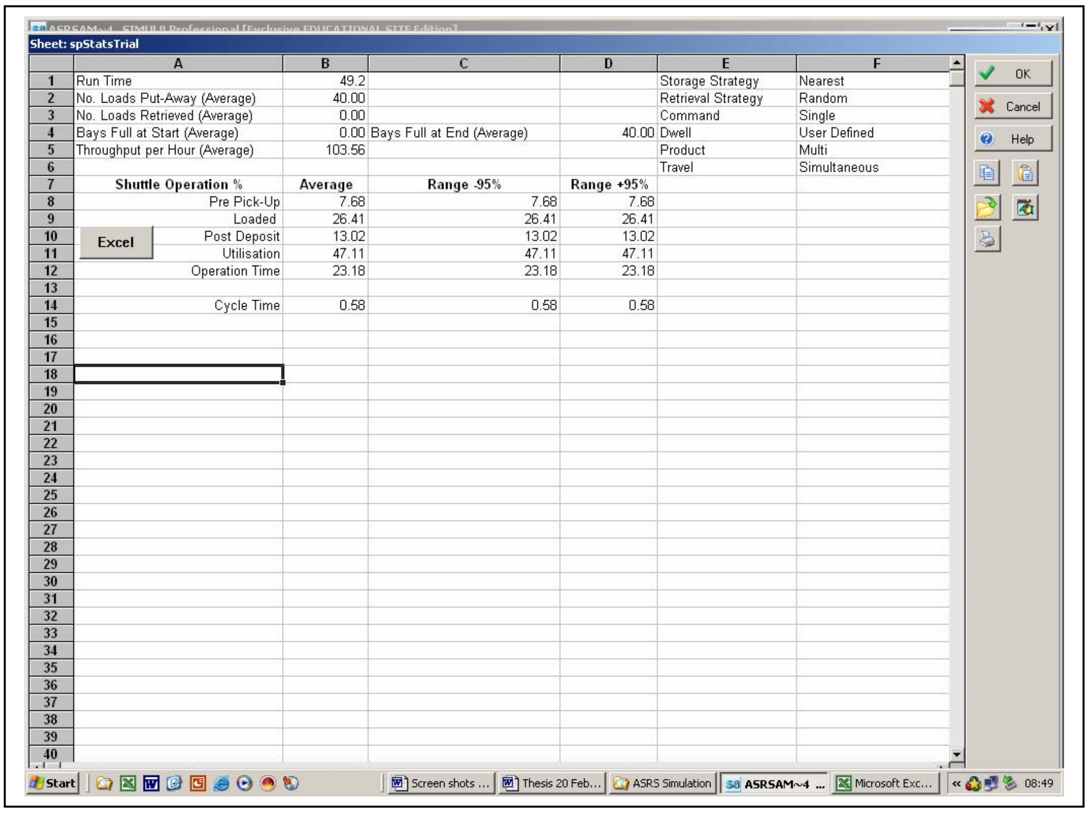

ASRSAM~4 SIMUL8 Professional [Exclusive EDUCATIONAL SITE Edition]
File Edit Clock Trials Results Objects Graphics Tools Control Finance Help
Clock
Conducting
Build Tools (2)
SIMUL8 Results Summary
Detail
Help
OK
Information
Educational Site License
Start

Results

		Low 95% Range	Average Result	High 95% Range
ShuttlePre	Working %	7.68	7.68	7.68
ShuttlePost	Working %	13.02	13.02	13.02
ShuttleLoaded	Working %		26.41	26.41
QCollect	Items Enterec		40.00	40.00
QNoStore	Items Enterec		0.00	0.00
ASRS Output	Number Com		0.00	0.00
Unprocessed Demand	Items Enterec		0.00	0.00
StoragePoint	Blocked %		0.00	0.00
Bay 1 1	Blocked %	73.75	73.75	73.75
Bay 1 2	Blocked %	80.51	80.51	80.51
Bay 1 3	Blocked %	82.90	82.90	82.90
Bay 1 4	Blocked %	85.29	85.29	85.29
Bay 1 5	Blocked %	84.10	84.10	84.10

Blank Row is 99 and Trial is 85

Figure 5-6: Output for Spreadsheet



	A	B	C	D	E	F
1	Run Time	49.2			Storage Strategy	Nearest
2	No. Loads Put-Away (Average)	40.00			Retrieval Strategy	Random
3	No. Loads Retrieved (Average)	0.00			Command	Single
4	Bays Full at Start (Average)	0.00	Bays Full at End (Average)	40.00	Dwell	User Defined
5	Throughput per Hour (Average)	103.56			Product	Multi
6					Travel	Simultaneous
7	Shuttle Operation %	Average	Range -95%	Range +95%		
8	Pre Pick-Up	7.68	7.68	7.68		
9	Loaded	26.41	26.41	26.41		
10	Post Deposit	13.02	13.02	13.02		
11	Utilisation	47.11	47.11	47.11		
12	Operation Time	23.18	23.18	23.18		
13						
14	Cycle Time	0.58	0.58	0.58		

Figure 5-7: spStatsTrial

5.3 Model Trials and Results

5.3.1 Model Test Conditions / Data

The ASRS developed in W.I.T. is configured as shown in Table 4. The horizontal travel speed is 0.078 m/s and the vertical travel speed is 0.049 m/s.

Table 4: Bay Co-Ordinates

Col.(mm)	0	376	577	819	1019	1261	1462	1664	1906	2107	2309	2550
Row(mm)	0	196	481	801	1044	1284						

The Pick Point is located at (4,3) and the Deposit Point is located at (5,3). Bays located at (4,2) and (5,2) are unavailable for use because of the load and unload conveyor design.

Trials consist of three runs with results averaged over this number of runs and a ± 99% Confidence Interval (CI) also reported. Three different sets of Requirements were placed on both models. Each of these sets consisted of a total of 40 operations, chosen so that each run was for approximately 20 minutes thus ensuring that, on average, 3 runs could be completed every hour:

- 40 Store operations
- 23 Store Operations / 17 Retrieve Operations
- 20 Store Operations / 20 Retrieve Operations

Trials A, B and C were conducted on both models with a product mix consisting of ten part numbers (Product ID's 100 – 109).

A summary of the results obtained for each of these conditions is presented in the following section with a brief summary of the strategies implemented for each trial.

5.3.2 Test Results

The following three separate sets of trials were conducted:

- Trial A: 40 store operations
- Trial B: 23 store / 17 retrieve operations
- Trial C: 20 store / 20 retrieve operations

Trial A

- 40 store operations

- All bays empty at start of trial
- S/R machine at Dwell point at start of trials
- 3 classes defined
- Bays dedicated in groups (Bay 1 – Bay 7 100, Bay 8 – Bay 14 101, etc.)

Table 5: Results - 40 Store Operations

				Math. Model		Phys. Model		Diff (%)
Trial No	**Dwell Point**	**Storage**	**Retrieval**	**Avg (Sec)**	**Range (Sec)**	**Avg (Sec)**	**Range (Sec)**	
1	User def. @ 6,3	Free - R	N / A	40.6	25	38.2	71	5.9
2	User def. @ 6,3	Free - N	N / A	34.8	N/A	35.3	7	-1.6
3	User def. @ 6,3	Class - R	N / A	38.9	70	37.9	18	2.5
4	User def. @ 6,3	Class - N	N / A			34.2	4	
5	User def. @ 6,3	Ded. - R	N / A	37.3	45	35.6	52	4.6
6	User def. @ 6,3	Ded. - N	N / A			34.9	86	6.5
7	Current	Free - R	N / A	36.5	100	37.3	40	-2.2
8	Current	Free - N	N / A	29.1	N/A	31.2	6	-7.3
9	Deposit Point	Free	Random	38.2	38	37.1	38	2.9
10	Deposit Point	Free	Nearest	31.4	N/A	31.5	28	-0.4

Main Findings

- 'Random' results will always be higher or at best equal to 'Nearest' results.
- SIMUL8 model not programmed for 'Class Nearest' or 'Dedicated Nearest' storage strategies.
- Range of results in 'Random' expected due to inherent properties of randomly choosing a bay location.
- Results for 'Current' dwell point show a 16.5% improvement from a 'User Defined' dwell point at (6,3)
- Results for 'Deposit point' dwell point show a value very close to 'Current' dwell point. This is due to the movement of the S/R machine being very similar for these two strategies, either returning to (6,3) or (5,3) before moving to (4,3) to pick a new pallet.
- Results for 'Class Nearest' show a 3% improvement than 'Free Nearest'. This requires further investigation.
- Results for 'Dedicated Nearest' show a negligible improvement from 'Free Nearest'. This is due to the designation of the bays in a 'quasi' random order.

- The correlation between the models varies between 0.4 – 7.3%. This is as expected for 'Random' strategy, however for the 'Nearest' trials (6 and 8) further work will need to be performed to improve the correlation and to gain a better understanding of this variation. This work will incorporate the S/R machine acceleration and deceleration profiles and the precision of the positioning of the S/R machine at the bays among other considerations.

Trial B

- 23 store / 17 retrieve operations
- All bays empty at start of trials
- S/R machine at dwell point at start of trials
- 3 classes defined
- Bays dedicated in groups (Bay 1 – Bay 7 100, Bay 8 – Bay 14 101, etc.)

Table 6: Results - 23 Store / 17 retrieve Operations

						Math	Phys	
Trial No	**Dwell Point**	**Storage**	**Retrieval**	**Travel**	**Command Type**	**Avg (Sec)**	**Avg (Sec)**	**Diff (%)**
1	User def. @ 6,3	Free - R	Random	Sim.	Single	35.8	39.6	-10.8
2	User def. @ 6,3	Free - N	Nearest	Sim.	Single		28.6	
3	User def. @ 6,3	Free - R	Random	Sim.	Dual	33.1	28.4	14.3
4	User def. @ 6,3	Free - N	Nearest	Sim.	Dual		22.1	
5	User def. @ 6,3	Free - R	Random	Rect.	Single	49.1	46.0	6.2
6	User def. @ 6,3	Free - R	Random	Rect.	Dual	41.7	36.2	13.2
7	Current	Free - R	Random	Sim.	Single	29.5	31.6	-6.9
8	Current	Free - N	Nearest	Sim.	Single		22.0	
9	Current	Free - N	Nearest	Sim.	Dual		18.3	
10	Pick Point (4,3)	Free – R	Random	Sim.	Single	35.0	35.9	-2.6
11	Pick Point (4,3)	Free – N	Nearest	Sim.	Single		23.9	
12	Pick Point (4,3)	Free – N	Nearest	Sim.	Dual		18.5	
13	Origin	Free - R	Random	Sim.	Single	52.3	62.5	-19.4
14	Origin	Free - N	Nearest	Sim.	Single		47.8	
15	Origin	Free - N	Nearest	Sim.	Dual		32.7	

Main Findings

- 'Random' results ≥ 'Nearest' results.
- Mathematical model not programmed for simulating 'Nearest' retrieval.

- ‘Dual’ command shows an average of 23.5% improvement from ‘Single’ command.
- The greatest improvement from ‘Single’ to ‘Dual’ occurs with a Dwell point at the origin (where travel of the S/R machine is greatest).
- The smallest improvement from ‘Single’ to ‘Dual’ is 16.9% with a ‘Current’ Dwell point (where travel of the S/R machine is least).
- ‘Rectilinear’ travel shows a 16% dis-improvement from ‘Simultaneous’ travel (‘Random result).
- Best performance observed for: Dwell point – ‘Current’, ‘Free Nearest’ storage, ‘Nearest’ retrieval, ‘Simultaneous’ travel and ‘Dual’ command. Values for Dwell point at the ‘Pick point’ were very similar.
- These show an improvement of 16% from the user defined position (6,3)

Trial C

- 20 store / 20 retrieve operations
- All bays full at start of trials according to the following patterns:
    - Trials 1 – 11: in series Bay 1 – 100, Bay 2 – 101, etc
    - Trials 12 – 22: Bay 1 – Bay 7 100, Bay 8 – Bay 14 101, etc.
- S/R machine at dwell point at start of trials
- 3 classes defined
- Bays dedicated in groups (Bay 1 – Bay 7 100, Bay 8 – Bay 14 101, etc.)

Main Findings

- Series of trials show similar good results (trials 2, 3, 6, 7, 8, 9 & 11). These results are influenced by the fact that all the bays were full at the start of the trial and therefore a retrieve (‘Nearest’) had to occur initially.
- Trial 8 (‘Dual’) has similar conditions to trial 7 except that all dual operations are in the sequence S – R with a slight reduction in the result.
- Trial 14 has similar conditions to trial 13 except that all dual operations are in the sequence S – R with a significant reduction in the result.
- Results for the Dwell point at (0,0) are better than those in Trial B because retrieval is from the ‘Nearest’, which in this scenario is the nearest to the start position of the S/R machine i.e. (0,0).

- ‘Dual’ results are on average 23% better than ‘Single’ results.
- ‘Class’ based strategy performs better than ‘Dedicated’ strategy.

Table 7: Results – 20 Store / 20 Retrieve Operations

						Math	Phys	
Trial No	**Dwell Point**	**Storage**	**Retrieval**	**Travel**	**Command Type**	**Avg (Sec)**	**Avg (Sec)**	**Diff (%)**
1	Current	Free - N	Nearest	Sim.	Single	N/A	24.4	
2	Pick Point	Free - N	Nearest	Sim.	Single	N/A	23.1	
3	Deposit Point	Free - N	Nearest	Sim.	Single	N/A	23.0	
4	Origin	Free - N	Nearest	Sim.	Single	N/A	41.4	
5	User def @ (6,3)	Free - N	Nearest	Sim.	Single	N/A	25.1	
6	Current	Free - R	Nearest	Sim.	Dual	N/A	23.5	
7	Pick Point	Free - N	Nearest	Sim.	Dual	N/A	23.3	
8	Pick Point	Free - N	Nearest	Sim.	Dual	N/A	24.1	
9	Deposit Point	Free - N	Nearest	Sim.	Dual	N/A	23.3	
10	Origin	Free – N	Nearest	Sim.	Dual	N/A	27.8	
11	User def @ (6,3)	Free - N	Nearest	Sim.	Dual	N/A	22.3	
12	User def @ (6,3)	Free - N	Nearest	Sim.	Single	N/A	26.9	
13	User def @ (6,3)	Free - N	Nearest	Sim.	Dual	N/A	24.0	
14	User def @ (6,3)	Free - N	Nearest	Sim.	Dual	N/A	28.0	
15	User def @ (6,3)	Ded - R	Nearest	Sim.	Single	N/A	30.7	
16	User def @ (6,3)	Ded - R	Nearest	Sim.	Dual	N/A	27.0	
17	User def @ (6,3)	Ded - R	Random	Sim.	Single	38.7	36.0	7.0
18	User def @ (6,3)	Ded - R	Random	Sim.	Dual	37.6	31.7	15.7
19	User def @ (6,3)	Class - R	Nearest	Sim.	Single	N/A	29.7	
20	User def @ (6,3)	Class - R	Nearest	Sim.	Dual	N/A	25.8	
21	User def @ (6,3)	Class - R	Random	Sim.	Single	38.2	34.0	11.0
22	User def @ (6,3)	Class - R	Random	Sim.	Dual	36.9	32.0	13.3

See Appendix F for a complete listing of the SIMUL8 ASRS model results, these results are shown as outputted from the model to a MSExcel spreadsheet. This model, as already described, allows for the inputting of requirements (both Storage and Retrieval) from a Requirements table in the ASRS Database which is used to control the physical ASRS.

Examination of this output file shows that the details of each trial are listed off in the relevant column. This includes: Trial number (next available number in next available row), Date, Storage Strategy, Retrieval Strategy, Command Type, Dwell Point, Product Range, Travel Type, Racking Configuration, S/R Machine Speeds, Storage Requirement Interval, Demand Requirement Interval, Warm-Up Period (seconds), Results Collection Period (seconds), Data Set (Base Random Number Set), Trial Details, Average Cycle Time, Throughput / Hour, Comment, Run Time, Number of Loads Put-Away, Number of Loads Retrieved, Bays Full at Start, Pre-Pick-Up (%), Loaded (%), Post Deposit (%), Utilisation and Operation Time.

There is a great need for much further research to be carried out using both models to determine the performance of an ASRS depending on the varying requirements encountered.

5.4 Summary

This chapter of the thesis summarises the mathematical ASRS model generated in SIMUL8. It explains the range of strategies which can be tested plus the various types of requirements which can be handled. The one extremely important feature of this model is its capacity to import its Storage and Demand values from a MSAccess Database defined over an ODBC link. This database can also be used as the controlling database for the physical ASRS. This allows total correlation between the requirements on both models. This is very important in proving both and gives the user greater confidence in the results from both. Appendix F lists off the trial results from this model, in summary, however the findings show very good correlation between both models.

The findings from both models indicate that the ASRS performs best when the following conditions are met:

Dwell Point: Pick Point, Deposit Point or Current Location

Travel Type: Simultaneous

Storage Strategy: Free – Nearest

Retrieval Strategy: Nearest

More research will need to be conducted to determine the ASRS performance across a wide range of demands.

The next chapter will comment on the research conducted in the context of a knowledge creation case study based on the dynamic knowledge creation model proposed by Nonaka et al (1994).

6 Dynamic Knowledge Creation Case Study

6.1 Introduction

One of the main objectives of this research was to develop a deep understanding of Automatic Storage and Retrieval Systems and capture and elaborate on this as a case study in knowledge creation.

As already stated in Chapter 4 the development of this ASRS underwent a number of significant steps from its initial build. These steps are:

- Initial programming of specific functions by O' Mahoney (2004)
- Specific strategy related PLC functions
- Manual control of S/R machine and forks
- Button activated SCADA Scripts
- SCADA and PLC programs for specific requirements and locations
- Generic PLC functions
- Integration of database communication into SCADA programming
- Development of SCADA project functions and actions
- SCADA and PLC programs for a variety of strategies
- Integration of more advanced strategies into script
- Integration of project functions and global actions in SCADA script
- Development of database to ensure the re-evaluation of trial data
- Development of Order Generator
- Development of mathematical model
- Trials
- Further program development (Dual command at location)
- Trials
- Timing of calculations changed in data control
- Further trials

In this chapter, the capture and creation of knowledge during these phases is outlined using the Nonaka model [Nonaka, 1994] outlined in Chapter 2 (Section 2.14).

It is the intention of the author to outline the development of the test-bed as a case study in knowledge creation with the intention of applying the experience to other projects in the future. The emphasis during the chapter will be on summarising the

various elements of this model and then linking the work completed to each of these steps, where possible.
A number of undergraduate projects were initiated and controlled by the author. The interaction between the previous postgraduate student, the six undergraduate students, the author and the staff in WIT is a major factor in the research environment that will also be analysed in this chapter.

6.2 Knowledge Creation Model

The Knowledge Creation model proposed by Nonaka et al. (1994) suggests that dynamic knowledge creation consists of three elements:

1. The SECI process: The process of knowledge creation via conversion from tacit to explicit knowledge
2. ba: The shared context for knowledge creation
3. Knowledge assets: The inputs, outputs and moderators of the knowledge-creating process

These three elements of knowledge creation have to interact with each other to form the knowledge spiral that creates knowledge.

6.2.1 The SECI Process

An organisation creates knowledge by means of the interactions between explicit knowledge and tacit knowledge. This interaction is called "knowledge conversion". In this conversion process, tacit and explicit knowledge expand in both quality and quantity. The four modes of knowledge conversion are:

- Socialisation: conversion from tacit knowledge to tacit knowledge
- Externalisation: conversion from tacit knowledge to explicit knowledge
- Combination: conversion from explicit knowledge to explicit knowledge
- Internalisation: conversion from explicit knowledge to tacit knowledge

Socialisation:
This is the process of converting new tacit knowledge through shared experiences.

Throughout the course of this research both the main researcher and a number of undergraduate students have been exposed to quite intensive hands-on experience in conjunction with previous researchers in this area. The benefit of the researchers

spending time together on this project has ensured a transfer of tacit knowledge between them over the last four to five years. It has also helped enormously that the main researcher has remained in situ for this period and has taken on the role of supervisor of the other undergraduate researchers.
This mode is seen during the majority of the stages outlined in Section 6.1, especially those where there was a high degree of interaction between the students and the author. A large part of the initial programming, both of the PLC and of the button activated SCADA scripts were a collaboration between researchers. Later on when the author's tacit knowledge base was increased, the flow of tacit knowledge was in a large part uni-directional to the students.

Externalisation:
Externalisation is the process of articulating tacit knowledge as explicit knowledge. This allows for the sharing of knowledge and becomes the basis of new knowledge.

Throughout the course of this research project great emphasis has been placed on the recording and documentation of all the researchers work and findings. All code developed has been thoroughly documented and a number of reports have been produced outlining the progress of the researchers. This has allowed for a smooth transfer of knowledge between the various personnel involved.
This mode is most clearly evident in the on-going progress reports and theses produced by O' Mahoney (2004), Coffey & Buggy (2004), McKenna & Naughton (2005) and Shouldice & Kent (2006). This documentation has been invaluable in the transfer of knowledge to future students.

Combination:
This is the process of converting explicit knowledge into more complicated and systematic sets of explicit knowledge.

The creation of this thesis plus undergraduate research project theses help to combine explicit knowledge from both inside and outside the organisation to form new knowledge.

Internalisation:
Internalisation is the process of embodying explicit knowledge as tacit knowledge.

Via internalisation, explicit knowledge created in the previous step (Combination) is shared throughout the manufacturing research community in the College and converted into tacit knowledge by individual students and researchers. This step in the knowledge creation process involves the new researchers analysing and absorbing the knowledge presented to them in explicit form and combining it with their own existing internal tacit knowledge helping to add a deeper understanding of the process being studied, knowledge being added to knowledge gained through different experiences and backgrounds.
This mode has been most clearly seen in the way that undergraduate students have progressed their own final year projects. Initially there is an over-whelming effect on the students as they attempt to come to terms with the vast amount of knowledge they are exposed to. Gradually they become familiar with this knowledge and assimilate it as internal knowledge and then build on this knowledge to advance their own specific project.

6.2.2 ba: The Shared Context for Knowledge Creation
Knowledge needs a context to be created. Nonaka et al. (1994) state that the knowledge-creating process is necessarily context-specific in terms of who participates and how they participate. Knowledge needs a physical context if it is to be created. ba offers such a context. It is created by means of the interactions among individuals or between individuals and their environments, rather than by an individual acting alone. A close physical interaction is important in sharing the context and forming a common language among participants.
ba is a place (specific time and space) where new knowledge is created.
According to Nonaka et al. (1994) there are four types of ba:
1 originating ba
2 dialoguing ba
3 systemizing ba
4 exercising ba
These are defined by two dimensions:

Type of interaction – that is, whether the interaction takes place individually or collectively.
Media used in such interactions – face-to-face contact or virtual media

Originating ba:
This is defined by individual and face-to-face interactions.

This is the place where tacit knowledge is shared (socialisation) and in the context of this research it is the on-going contact between researchers either directly involved in the ASRS upgrade project or those connected to the AMT research group. The face-to-face interactions are essential in enabling the transfer of the tacit knowledge between individual lecturers. Working through problems in combination is a very efficient way of increasing ones knowledge base and so it has proved during this project.

Dialoguing ba:
This is defined by collective and face-to-face interactions. It is the place where individual's mental models and skills are shared, converted into common terms and articulated as concepts. This is the perfect forum for externalisation. The tacit knowledge among participants is shared openly through constructive dialog.

Within the context of this research project this dialoguing ba consisted of the researchers meeting in formal groups with the lead researchers in the Manufacturing sector. This gave the project researchers an opportunity to formulise their thoughts and present their findings in a semi-formal forum and receiving immediate feedback. Individual's tacit knowledge is shared and articulated through dialogues among participants. Each individual at the meeting gains in knowledge immediately and there is also the added benefit of further expansion in knowledge through self-reflection (analysing the newly acquired knowledge within the context of their own individual knowledge and experiences).

Systemizing ba:
This type of ba is defined by collective and virtual interactions. This provides a context for the knowledge conversion mode of combination. Modern communication

tools such as on-line networks, e-mails, databanks and newsgroups allow for the dissemination of knowledge and information effectively and efficiently.

Throughout the course of this project there has been extensive use of the GroupWise e-mail system to inform all interested personnel on the on-going progress and to acquire their input into the project.

Exercising ba:
Exercising ba is defined by individual virtual interactions and offers a context for the internalisation of knowledge. This is where the researcher embodies explicit knowledge that is communicated via virtual media such as written manuals, theses or simulation programs.

Over the course of this project a number of undergraduate students have internalised knowledge made available to them from previous undergraduate students by way of documented programs and descriptive theses written at the end of the particular student's time on the project. This has been invaluable to the new students and has been very beneficial to the project.

The interaction of different types of ba in the process of knowledge creation:
For any group or organisation intent on creating knowledge it is important that procedures are put in place which support and compliment the development of the various types of ba. These procedures can include the deliberate use of face-to-face interactions in an Originating ba environment for all personnel involving extensive on-the-job training on the shop floor. The tacit knowledge gained in this phase of the knowledge cycle is converted into explicit knowledge in the form of 'hypotheses' about the topic (research, market needs, production issues, etc.). Within a research environment these hypotheses are freely tested by the individual researchers, allowing for their further refinement.

6.2.3 Knowledge Assets

Knowledge assets form the basis of the knowledge-creating process. These assets may be firm-specific resources that are indispensable to creating value for the firm. It is very difficult to evaluate and manage knowledge assets, unlike most other commodities they cannot be bought and sold. They are also constantly evolving as a result of the organisations informal or formulised knowledge creation system. Nonaka et al. (1994) suggest four categories or types of knowledge assets:

1 experiential
2 conceptual
3 systemic
4 routine

Experiential Knowledge Assets

These consist of shared tacit knowledge, which is built by means of shared, hands-on experience among the members of the group or organisation. Examples of this type of knowledge asset include: skills, know-how, care, trust, facial expressions, gestures, enthusiasm, tension and improvisation.

All the researchers involved in this project developed programming skills in MSAccess, SIMUL8, WinCC and Step7 which they were able to build on through the course of their work.

Conceptual knowledge assets

These type of assets have tangible form making them easier to be grasped than experiential knowledge assets, although there is still a problem clarifying what members actually perceive.

This type of knowledge asset is best demonstrated by the over-riding concept influencing the project which was the development of an ASRS test-bed. This had an impact on all decisions made on design and programming. This was never just the development of an ASRS to service the existing FMS, if this was the case then many of the features in the finished design would not be there. An ASRS which is solely servicing an FMS would have a limited number of options on dwell point, retrieval and storage strategy, forks control and travel type. The concept of a ‘test-bed’ has progressed the ASRS development in an entirely new direction.

Systemic knowledge assets
These knowledge assets consist of systematised and packaged explicit knowledge, such as product specifications, manuals and documented and packaged information about customers and suppliers.

These assets are best demonstrated in the range of theses and reports published on this research.

Routine knowledge assets
These type of assets include that tacit knowledge that is routine and embedded in the actions and practices of the organisation including know-how, organisational routines and culture in carrying out the daily business of the organisation.

These knowledge assets, having being developed in the students are lost to the college when the student leaves. Very rarely are these assets documented, it is up to individual students to develop these for themselves through interaction with various lecturers, technicians, support staff and students and lecturers. The culture will even vary between different classes in the same year not to mention between the different years.

A report issued by the Organisation for Economic Co-Operation and Development [OECD, 1996] entitled "The Knowledge-Based Economy" also categorises the different types of knowledge which it is claimed are important in the knowledge-based economy: know-what, know-why, know-how and know-who. Know-what and know-why can be obtained through reading books, attending lectures and accessing databases, the other two types of knowledge are gained through practical experience. Know-how is typically learned in situations where an apprentice follows a master and relies upon him as the authority. Know-who is learned in social practice and sometimes in specialised educational environments and is also developed in day-to-day dealings with customers, sub-contractors and independent institutes. Know-who is socially embedded knowledge that cannot easily be transferred through formal channels of information.

6.3 Knowledge Expansion through Spiralling

According to Nonaka et al (1994) knowledge is dynamic as it is created in social interactions among individuals and organisations. Knowledge is context specific because it depends on a particular time and space. Without a context, it is just information, not knowledge. Knowledge is also humanistic, because it is essentially related to human action. Information becomes knowledge when it is interpreted by individuals and given a context and anchored in the beliefs and commitments of individuals. Hence, knowledge is relational – such things as 'truth', 'goodness' and 'beauty' are in the eye of the beholder.

There are two types of knowledge: explicit knowledge and tacit knowledge. Explicit knowledge can be expressed in formal and systematic language and shared in the forms of data, scientific formulae, specifications and manuals. It can be processed, transmitted and stored relatively easily. In contrast, tacit knowledge is highly personalised and hard to formalise. Subjective insights, intuitions and hunches fall into this category of knowledge. Tacit knowledge is deeply rooted in action, procedures, routines, commitment, ideals, values and emotions. It is difficult to communicate tacit knowledge to others, as it is an analogue process that requires a kind of 'simultaneous processing'.

Tacit and explicit knowledge are complementary and both types of knowledge are essential to knowledge creation. Explicit knowledge without tacit insight quickly loses its meaning. Knowledge is created by means of interactions between tacit and explicit knowledge, rather than from tacit or explicit knowledge alone.

The three elements of knowledge creation, SECI process, ba and knowledge assets, have to interact with each other to form the knowledge spiral that creates knowledge.

6.4 Knowledge Creation Findings

There have been a number of phases through which the ASRS test-bed has progressed:

Initially O' Mahoney (2004) with the author as supervisor designed and built the ASRS and developed very limited SCADA based control.

The next phase involved the author supervising three groups of under graduate students in the development of basic button operated SCADA scripts and specific strategy related PLC functions (approximately forty of these being developed by Shouldice & Kent (2006)).

After this the author developed, generic PLC functions, to allow for greater control flexibility, and database communication, which were both included in button operated SCADA.

The next phase of the development by the author was the writing of SCADA project functions and global actions that also included advanced strategies for ASRS control.

The integration of the control database, the order simulator and the mathematical model came next and this allowed the author to conduct a number of trials which highlighted some control problems with the system.

Further program development and trials followed. A lack of correlation between the physical and mathematical models led to further program development and finally more trials.

During each of these phases there has been interactions leading to the creation of knowledge assets, albeit at a very low level, within the knowledge environment of the AMT laboratory of WIT.

Initially explicit knowledge was accessed by O' Mahoney (2004) and the author on current ASRS research. This explicit knowledge being available through various books and articles on the subject and also through reports made available by the partnership company. Explicit knowledge was also provided by the suppliers of the two main software packages, this being more thoroughly assimilated through attendance at specific training courses on the software where the explicit knowledge was shared tacitly in a 'learning by doing' environment.

In the originating ba environment of designing and building a particular machine the internalisation mode of knowledge conversion dominated, both researchers developing a broad understanding of the hardware, software and the application.

Through progress reports and development group meetings an externalisation process developed culminating in the production of a Masters thesis by O' Mahoney (2004).

The author then embarked on a series of SECI processes with groups of undergraduate students, ending with the development of specific strategy related PLC functions and button activated SCADA scripts. The process varying or developing slightly from year to year as the author became more familiar with the hardware and software and also the knowledge process that was at play.

This resulted in better program documentation and more detailed progress reports and final theses. The process of passing on of tacit knowledge from the author to the

students also improved, possibly as the authors tacit understanding of the system increased.
The final phase in the development of the system has seen the author develop an even deeper understanding of the systems through a further externalisation, combination and internalisation process resulting in a series of trials which have helped to refine the system further.

6.5 Summary

Throughout the course of this research great emphasis has been placed on the retention, transfer and creation of knowledge and the development of an environment where this could occur. The fact that the chief researcher / author was also a full-time lecturer in the organisation greatly facilitated this process. Researchers were encouraged to increase their own tacit knowledge in an environment created by the research group, the opportunity was also there for researchers to develop their hypotheses through analysis and testing. Knowledge was retained and passed on to other undergraduate researchers through a well-developed documentation process and the production of specific knowledge based theses.
The most significant developments from this were the rapid progress made both by the undergraduate researchers and the physical development of an actual ASRS test-bed.
Examples of the documented programs are shown in Appendix A and a range of theses written by both postgraduate and undergraduate students demonstrate the wealth of explicit knowledge generated.
It is the intention of the author to apply the information accumulated from this knowledge case study to other broader applications within WIT.

7 Conclusion & Recommendations

7.1 Introduction

The main goal of this project was to develop a test-bed for ongoing research in ASRS design allowing for the analysis of a variety of control strategies both mathematically and empirically within a knowledge paradigm of knowledge acquisition, tacit knowledge development, tacit knowledge extension, explicit knowledge extension and knowledge spiralling. This chapter reviews the various areas of work carried out as described in Chapters 3 to 6, in terms of the outputs achieved. Overall conclusions (Section 7.4) are presented, examining the value of the work in relation to the objectives and also in terms of the future evolution of the system. Finally recommendations are set out for each area of the project's development, suggesting improvements and enhancements to bring the ASRS test-bed to the next level of development.

The major benefits of development of both a SIMUL8 model and physical model include:

- Better understanding of ASRS control
- Realisation that the timing of calculations is important in the physical model
- Sequence of operations is obvious in a physical model
- Proofing of results in SIMUL8 model
- Clarification of strategies to interested parties
- Knowledge gained on the interaction of dwell point to retrieval strategy

In general, dual control improves performance (in terms of throughput), simultaneous travel is better than rectilinear travel, dwell point at origin gives very poor results, dwell point at current, pick point or deposit point is best. The best performance was observed with current dwell point, simultaneous travel, dual control, free-nearest storage and nearest retrieval strategies. Similar excellent results were observed with pick point and deposit point dwell strategies. There is little difference in dual control storage / retrieval or retrieval / storage operation. The results for FIFO and random retrieval strategies were very similar. The SIMUL8 ASRS Simulation model shows greatest correlation with: dwell point, random storage and retrieval. The best mathematical results were also achieved with the above strategy mix.

In terms of creating a knowledge environment the development of the physical model was hugely beneficial in the gaining of tacit knowledge by the chief researcher and supervisor and the development of a number of undergraduate students (3 groups) plus a realisation that the greater the volume of tacit and explicit knowledge available the faster the up-take for the students. There are also great benefits to the College research centre of having a full time researcher in this area, especially when the subject matter becomes complex and where the learning curve for students can be very steep. There was previously a loss of knowledge in both tacit and explicit form (most noticeably from the postgraduate researcher). It was difficult to disseminate and log this knowledge and an in-house expert is very valuable. The hands-on involvement of the chief researcher allows for faster learning for all and more focussed research for undergraduate students. Knowledge reached a critical mass and then progression, especially for chief researchers was rapid. Documentation is crucial and a learning program for students is recommended. The knowledge gained by undergraduate researchers was almost entirely retained due to the involvement of the chief researcher and supervisor. The chief researcher also passed this knowledge on to other undergraduate students by incorporating the recently gained knowledge into lecture material.

7.2 Summary of Aims and Objectives

The aims and objectives set down in Chapter 1 were as follows:

1. Develop an understanding of industrial storage and retrieval systems.
2. Develop fieldbus based control software and integrate this hardware with a central database and an order generator.
3. Develop an entirely flexible ASRS control system which would allow for the empirical testing of a wide range of control strategies incorporating a mathematical model to generate a broad spectrum of requirements in terms of parts, operation and time required.
4. Utilise a mathematical model of the ASRS in combination with output to a results spreadsheet to confirm the findings from the physical model and to produce long term data on performance etc.
5. Carry out extensive testing, troubleshooting and trials on both models reviewing the results and making recommendations on most suitable settings for a variety of requirements on the ASRS.
6. Develop a deep understanding of the existing ASRS and capture and elaborate on this as a model in knowledge creation.

7. Showcase to the sector the benefits of the knowledge creation process as a practical solution to the on-going problem of poor retention of knowledge assets.

7.3 Results

7.3.1 ASRS Mechanical and Electrical Design & Fabrication

The mechanical and electrical design and fabrication process is outlined in Chapters 3 and 4 of this thesis. Great emphasis was placed on the design of a system that would allow for flexibility in control, facilitating the testing of a range of parameters into the future. The knowledge base available in a local partnership company was also tapped into for this process and major steps were taken to ensure that the system installed in WIT accurately resembled the Hi-Bay system there albeit at a much smaller scale. The hardware chosen: Siemens S7, Profibus DP network, remote I/O unit, inverters on both AC motors, dual direction forks and the S/R machine design are all similar to the industrial unit in the partnership company.

WIT already had quite a knowledge base in Mitsubishi PLC control and Wonderware In-Touch SCADA software, however the partnership company's knowledge was centered on Siemens hardware and software. This had a major bearing on the decision to go with Siemens. The outcome of this has been to increase the knowledge base of WIT in a broader range of PLC control and SCADA software. Hence the manufacturing research group has now the capacity to make an informed decision on the benefits or otherwise of both of these systems. This knowledge is also being passed on to undergraduate students both through specific modules such as PLC control and SCADA communications and through research based final year degree program projects.

The hardware design chosen allows for the testing of: X and Y axis speed variation, rectilinear and simultaneous travel, dual rack design, forks on or off control, various sequences of store and retrieve functions, various dwell points, including the origin position.

A variety of safety and protection devices are included in the design to protect the operators (strategically positioned emergency stops) and the machine itself (over-travel sensors on all axes), sensors on the forks to ensure the S/R machine is in the correct position when the forks operate and a facility to isolate all three motors thereby allowing the PLC to switch on the outputs without actually powering these outputs). These have all proved invaluable during the course of the research and have also been refined as the project progressed.

7.3.2 ASRS Control

The programming and display screens developed with Step7 and WinCC are outlined in Chapter 4. The Step7 programs consist of one organisation block (OB1) and thirteen functions, four of which are used in full automatic control, the other functions are used for set-up and manual control. An important feature of the control is that the main decisions and intelligence is built in to the SCADA script with the PLC code only performing very basic functions. Initially this was not the case and extremely complex PLC code was produced with separate functions programmed for each strategy. At present a total of 11520 individual strategies can be selected. This figure comprises: Dwell Point (5 options), Travel Type (2 options), Command Type (2 options), Control Type (2 options), Fork Control (2 options), Pre-Store Buffer (2 options), Pre-Retrieve Buffer (2 options), Obsolete Zone (2 Options), Store Strategy (6 options) and Retrieve Strategy (3 options). It was quickly discovered that the original plan of individual programs or functions for each strategy was not an option and the decision was taken to simplify this code and use a combination of WinCC tools to control the system. These tools include: WinCC code behind individual buttons setting various tags high and low depending on the status of certain inputs both external and internal to WinCC (‘Home’, ‘GoTo X’, ‘GoTo Y’, ‘GoTo Dwell’, ‘GoTo Home’, ‘Forks Out’, ‘Start’, etc.), Global script developed project functions (‘ASRSOp’, ‘SetPlaceCoOrds’, ‘ASRSOpComp’, ‘ASRSDwell’) and actions (‘GoToXYCompleteAction’, ‘PickComplete’, ‘PlaceComplete’, ‘PlaceComplete2’). The S/R machine travel is accomplished using two quite short Step7 functions (FC 17 and FC 18), these are called from a number of SCADA script project functions and global actions and return to a global action script (‘GoToXYCompleteAction’) every time, with the intelligence built into these scripts. When either FC 17 or FC 18 is called the two tags that are used by these functions (MW755, ‘xAUTOReqPosn’ and MW735, ‘yAUTOReqPosn’) are fixed by the calling SCADA script and the PLC function compares them with the two memory words storing the actual X and Y positions (MW745, ‘xAUTOActPosn’ and MW725 ‘yAUTOActPosn’).

The design of the control system in this manner allows for great flexibility and modularisation in the programming. It is therefore possible even in the future to include other more complex features without having to alter the PLC code. All the Step7 code is included in Appendix A and all the SCADA script is in Appendix C.
One very important finding discovered during the trials on this system was the delay in calculating the next required location (Bay No, X and Y co-ordinates). Initially this calculation was performed when the S/R machine was at rest, either at the Dwell Point after a Single operation or after the second of a Dual operation or at the place point after the first of a Dual operation. When this calculation was being performed (proportional to the number of available bays and the type of storage or retrieval strategy selected) the S/R machine was not moving. This lack of correlation showed up a problem with the physical model. By altering the control flow between the various PLC functions and the SCADA functions and actions this issue was sorted and the current situation is that there is no delay at the times specified above. This has improved the correlation between the two models.

7.3.3 Order Generator

This SIMUL8 developed program allows for the automatic creation of orders with the following factors capable of being controlled: Quantity of Requirements, Part Number, Operation Type and Operation Interval. A probability profile is supplied by the user for both the Operation Type and the Operation Interval. When this model is run the user is prompted to input the total number of orders required. Once the orders are generated the user is asked if the database linked through an ODBC link is to be updated.
The two tables that are then updated are 'tmprequirements' and 'interval times'. The updating of the database in this fashion allows the user to continuously re-use this set of requirements through a simple macro which:
modifies the contents of the bays
clears the completed jobs table
combines the 'tmprequirements' with the 'interval times' and the required start time (which the user is prompted to supply) to update the 'requirements' table.
During the testing of the ASRS many sets of requirements were generated using this model with each one then being utilised for numerous trials. To allow the testing and re-testing of various strategies with the same set of simulated requirements the

operator simply copies the relevant database into a back-up sub-directory and re-runs the model to generate a completely new set of requirements in the original copy of the database. See sections 7.3.5 and Appendix F for a complete set of results for the trials conducted from orders generated with this model

7.3.4 Database Control

The database used for this research is a MS Access database that is accessed from various other software through an ODBC link, 'ASRSDatabase'. This database consists of a number of tables: 'ASRS Contents', 'Requirements', 'Operations Completed', 'InterTimes' and 'tmpRequirements'. It also includes macros to implement start-up procedures and a number of reports and queries.

The communication between this database and the WinCC script is achieved using a Siemens WinCC add-on called WinCC DBExt-DLL. This program consists of a number of synchronous and asynchronous functions that allow a number of database requests including:

DBEGetVariables: Read data from a database table

DBESetVariables: Write data to a database table

DBEDeleteRecord: Deletes a database record from a database table

DBEGetNumber OfRecords: Get the number of records that matches the search condition

These, as explained in Chapter 4, require an expression, which returns a BOOL value depending on the success or failure of the request, and typically a variable structure or array into, or from which, field information for the relevant database record is placed.

The interaction between this database and the Order Generator developed in SIMUL8 is explained in 7.3.3. There is also interaction between this database and the ASRS model developed in SIMUL8. This is also conducted through the ODBC link, ASRSDatabase, and allows the SIMUL8 model to use the actual Requirements table as its input data. Thus allowing for complete correlation between the demands on both the physical and mathematical models.

The setting-up of this database and its interaction with the two SIMUL8 models and with the WinCC SCADA scripts has been very successful. There were a number of initial issues with the communication to SCADA, most noticeably the setting up of the SQL conditions and the timing of the updating of the variable structure. Chapter 4 outlines the steps taken to progress these issues.

7.3.5 Mathematical Model

A complete listing of the results obtained from the mathematical model are shown in Appendix F. This model exports the results and the settings to an open spreadsheet (ASRS Results.xls). This results summary includes: Results collection period (total run time), average cycle time, throughput / hour, number of loads put-away and retrieved, pre pick-up time, loaded time, post deposit time, utilisation and operation time. The settings summary includes: storage type, retrieval type, command type, dwell point, product range, travel type, racking size and travel speeds.

The results from the model show a high correlation with the results obtained from the physical model. The following is a summary of the findings from this model:

- Average cycle times for ' Free-Random' storage or retrieval strategies will always be higher or at best equal to cycle times for 'Free-Nearest' strategies due to the fact that 'Free-Nearest' strategies will always provide the shortest possible cycle times (i.e. the best possible 'Random' result).
- SIMUL8 model not capable of 'Class Nearest' or 'Dedicated Nearest' storage strategies.
- Range of results in 'Random' expected due to inherent properties of randomly choosing a bay location.
- Results for 'Current' dwell point show a 16.5% improvement from a 'User Defined' dwell point at (6,3)
- Results for 'Deposit point' dwell point show a value very close to 'Current' dwell point. This is due to the movement of the S/R machine being very similar for these two strategies, either returning to (6,3) or (5,3) before moving to (4,3) to pick a new pallet.
- Results for 'Class Nearest' show a 3% improvement than 'Free Nearest'. This requires further investigation.
- Results for 'Dedicated Nearest' show a negligible improvement from 'Free Nearest'. This is due to the designation of the bays in a 'quasi' random order.
- 'Dual' command shows an average of 23.5% improvement from 'Single' command.
- The greatest improvement from 'Single' to 'Dual' occurs with a Dwell point at the origin (where travel of the S/R machine is greatest).

- The smallest improvement from ‘Single’ to ‘Dual’ is 16.9% with a ‘Current’ Dwell point (where travel of the S/R machine is least).
- ‘Rectilinear’ travel shows a 16% reduction from ‘Simultaneous’ travel (‘Random result).
- Best performance observed for: Dwell point – ‘Current’, ‘Free Nearest’ storage, ‘Nearest’ retrieval, ‘Simultaneous’ travel and ‘Dual’ command. Values for Dwell point at the ‘Pick point’ were very similar.
- Some results are influenced by the fact that all the bays were full at the start of some of the trials and therefore a retrieve (‘Nearest’) had to occur initially.
- Results for the Dwell point at (0,0) are not as bad in Trial C as in Trial B because retrieval is from the ‘Nearest’, which in this scenario is the nearest to the start position of the S/R machine i.e. (0,0).
- ‘Dual’ results are on average better than ‘Single’ results.
- ‘Class’ based strategy performs better than ‘Dedicated’ strategy.

7.3.6 Trial Results

A complete listing of the results obtained from the physical model are shown in Appendix F. The results compare favourably with those obtained from the mathematical model. The findings outlined in 7.3.5 above are also valid for the physical model. Some interesting additional points discovered with the physical model:

1) The processing time required by the PC caused a delay in the S/R machine. This was subsequently corrected to allow off-line calculation and decision making, where practicable.
2) The finding that in dual command it did not seem to make any difference to average cycle times whether the majority sequence of operations was a store followed by a retrieve (traditional operation) or a retrieve followed by a store. This may be due to the ASRS being fully loaded which dictated the bay selection or else due to the ‘Dwell point’ strategy being implemented. Further research in this area could establish how general this finding is over a range of strategies.
3) That free-nearest retrieval does not always bring the S/R machine in the best direction, thus arriving at a situation where the S/R machine moves further and further away from the deposit point in deciding on the nearest retrieval bay.

When the dwell strategy is 'Current' or during the second operation in a 'Dual' cycle the storage or retrieval strategy 'Free-Nearest' may take the S/R machine in a direction which does not minimise total travel time. Further work is required to determine the overall effect of this with a view to evaluating the total travel time prior to deciding on the next bay.

4) When a required component is in the Bay where the S/R machine is currently positioned the program must allow the S/R machine to immediately pick that component rather than having to move to a different bay to pick.

7.3.7 Knowledge Management

The approach adopted during the course of this project has been to follow the dynamic creation model proposed by Nonaka et al. (1994). This model defines the spiral of knowledge as comprising of three elements: SECI, ba and knowledge assets. The four elements of SECI (Socialisation, Externalisation, Combination and Internalisation) combine to help the transformation of knowledge from Tacit to Explicit to Tacit. Within the confines of this project great emphasis has been placed on the development of knowledge through this process including the development, testing and dissemination of hypotheses. This has been accomplished by the close association between all researchers, past and present, the encouragement of researchers to 'try' things and the development of an environment where all researchers have an input into all current research, both formally and informally. This has been one of the major achievements of this research group. The development of a shared context for knowledge creation (ba) occurs almost by default in a research environment where the emphasis is placed on the development of a machine or system, in this case a fully flexible ASRS integrated with two SIMUL8 models. The majority of the interaction is by necessity individual and face-to-face but by careful control it has been possible to develop other types of contexts including collective face-to-face and virtual contexts. All four types of ba (originating, dialoguing, systemizing and exercising) have been used to good effect during the course of this project and have helped to define and develop a number of knowledge assets including very obvious experiential assets but also including conceptual, systemic and routine assets. It is hoped that the emphasis placed on knowledge creation during the course of this project will have an impact on future research in WIT and beyond.

7.3.8 Test-Bed Benefits

This project has resulted in an ASRS test-bed that will have major benefits to the users of hi-bay storage system in the future. The advantages of being able to test the impact of the complete range of strategies on a number of different requirement scenarios is limitless especially when fully integrated with a very flexible order generator and a mathematical model of the ASRS (which can be modified by an operator to model other much larger single bay storage systems).

It is essential that the research group make public the research asset that has been developed and encourage its use among the users of ASR systems in the local area and beyond. This will depend on the publication of future papers and the presentation of these at local, national and international conferences. There is no reason why WIT will not become a centre of excellence for ASRS and knowledge creation research.

7.4 Conclusions

The primary goal of this project was to develop a test-bed for on-going research in ASRS design allowing for the analysis of a variety of control strategies both mathematically and empirically within a knowledge paradigm: this has been achieved. The major benefit of developing the two models simultaneously is that the physical model can be used to gain knowledge on ASRS design and control and can also be used to prove the results from the mathematical model. The mathematical model can then be used to run more extensive trials on the various strategies. Initial trials suggest that optimum results are obtained with the following strategy mix: Dwell Point – Current, Movement – Simultaneous, Command Type – Dual, Storage – Free Nearest, Retrieval – Nearest. Further work with a variety of requirements and a mix of strategies is required with both models to reach more firm conclusions.

In terms of the objectives set out at the beginning of this project the following can be stated:

1. The primary goal to develop an entirely flexible ASRS control system that would allow for the empirical testing of a wide range of control strategies incorporating a mathematical model to generate a broad spectrum of requirements in the AMT Laboratory at WIT has been achieved (objective 3).
2. A thorough understanding of industrial storage and retrieval systems has been acquired by the chief researcher and a number of undergraduate researchers (objective 1).

3. The Control system developed is based on a Profibus-DP fieldbus network integrated with SCADA WinCC script, Siemens Step7 PLC code, a MSAccess database and a SIMUL8 based Order Generator (objective 2).
4. A SIMUL8 based Mathematical model of the WIT based ASRS with output to a MSExcel spreadsheet and input from the ASRS control database (ASRS1.mdb) has been developed (objective 4).
5. Extensive testing, troubleshooting and trials have been conducted on both the physical and mathematical models and a variety of initial findings have been summarised in this report (objective 5).
6. This project has been undertaken as an exercise in knowledge creation and several positive results from this have been highlighted in this report (objectives 6 and 7).

With regard to the future development of this system the following outcomes can be claimed for this project:

1. The integration of the SIMUL8 order generator with the ASRS control database and the further linking of this database to the ASRS schedule simulator allows for great flexibility in the range of strategies that can be tested both mathematically and empirically.
2. The development of a dynamic knowledge creation model based on this has great future benefits within WIT and beyond.
3. Great knowledge has been developed internally in WIT during the course of this project and this will have major benefits for a broad range of future students.

7.5 Recommendations

Great advances have been made to the ASRS through the course of this project especially in the flexibility of control and the in the development of the required hardware. Similar to many other such projects there is always room for improvement and development. These areas are summarised as follows:

- Develop and implement the link between the FMAS and the ASRS, allowing for the communication of requirements from one to the other.
- Conduct further trials on all strategies in both models to explore and improve correlation. Specifically, these trials will incorporate a study of the effect of

S/R machine acceleration and deceleration and also the precision of the S/R machine location at each bay.

- Conduct further trials on minimising total travel time with ‘Current’ dwell point, ‘Dual’ operation and ‘Free-Nearest’ retrieval.
- Conduct trials on the ASRS with a variety of requirements and help to establish the strategies best suited to a range of such requirements.
- Implement kitting and buffer zone control and determine the effect of this control on the overall efficiency of the ASRS.
- Further develop the SIMUL8 ASRS scheduling model to include the kitting and buffer zone control plus the inclusion of other storage and retrieval strategies.
- Utilise this knowledge creation case study developed during the course of this project in the other manufacturing based projects and possibly further a-field with the college research environment or beyond.
- Re-design of the fork mechanism to improve its robustness and also include a second parallel rack that will increase the capacity of the storage system and require further development of the control system and establish the flexibility of the existing fork design.
- Develop the MSAccess database or the MS SQL server database to provide a series of input screens for the operator (class type, initial ASRS Contents, changing of Contents etc.) and a selection of reports on Operations Completed, Requirements, Contents etc.

References

1. Alexander, R., *Engineering for Mixed Product Production in a Flexible Manufacturing and Flexible Assembly System,* WIT, (2005).
2. Badawy, M.K., *Technology Management Education: Alternative Models,* California Management Review, v40, p94-116, (1998).
3. Barry, J., *Engineering for Variety in a Fully Automated Multi Production Manufacturing System,* WIT, (Pending 2007).
4. Bettis, R.A. and Hitt, M.A., *The New Competitive Landscape,* Strategic Management Journal, v16, p7-19, (1995).
5. Boudreau, M., Loch, K.D., Robey, D. and Straud, D., *Going Global: Using Information Technology to Advance the Competitiveness of the Virtual Transnational Organisation,* Academy of management Executive, v12, p120-128, (1998).
6. Bozer, Y.A. and White, J. A., *Design and Performance Models for End-Of-Aisle Order Picking Systems,* Management Science, v36, n7, (1990).
7. Cardinal, L.B., Hatfield, D.E., *Internal knowledge generation: The corporate research laboratory and innovative productivity in the pharmaceutical industry*, Journal of Engineering and Technology Management, v17, p247-271, (2000).
8. Coffey, K. and Buggy, L., *Development of ASRS in WIT,* u/g project report, WIT, (2004).
9. Cheng, Y.T. and Van de Ven, A.H., *Learning the Innovation Journey: Order Out of Chaos?,* Organization Science, v7, p593-614, (1996).
10. Cohen, W.M. and Levinthal, D.A., *Absorptive Capacity: A New Perspective on Learning and Innovation,* Administrative Science Quarterly, v35, p128-152, (1990).
11. Chow, W-M., *An Analysis of Automated Storage and Retrieval Systems in Manufacturing Assembly Lines,* IIE Transactions, (1986).
12. Crosse, A., *Mechanical Integration of Programmable Machines with Short Cycle Flexible Manufacturing Cells,* WRTC, (1997).
13. Cyert, R. and March, J.G., *A Behavioral Theory of the Firm,* Prentice-Hall, (1963).
14. DeCarolis, D.M. and Deeds, D.L., *The Impact of Stocks and Flows of Organisational Knowledge on Firm Performance: An Empirical Investigation*

of the Biotechnology Industry, Strategic Management Journal, v20, p953-986, (1999).

15. Drucker, P.F., *Management Challenges for the 21st Century,* HarperBusiness, (1999).

16. Egbulu, P.J., *Framework for Dynamic Positioning of Storage/Retrieval Machines in an Automated Storage/Retrieval System,* International Journal of Production Research, v29, n1, p17-37, (1991).

17. Egbelu, P.J. and Wu, C.T., *A Comparison of Dwell Point Rules in an Automated Storage/Retrieval System,* International Journal in Production Research, v31, n11, p2515-2530, (1993).

18. Elsayed, E.A. and Lee, M.K., *Order Processing in Automated Storage/Retrieval Systems with Due Dates,* IIE Transactions, v28, p567-577, (1996).

19. Elsayed, E.A. and Unal, O.I., *Order Batching Algorithms and Travel-Time Estimation for Automated Storage/Retrieval Systems,* International Journal of Production Research, v27, n7, p1097-1114, (1989).

20. Fiol, M.C. and Lyles, M.A., *Organisational Learning,* Academy of Management Review, v10, p803-813, (1985).

21. Flanagan, B., *Dynamic Scheduling, Artificial Intelligence and Control in a Flexible Manufacturing System,* WIT, (2004).

22. Fowler, S. W., King, A. W., Marsh, S. J., and Victor, B., *Beyond products: new strategic imperatives for developing competencies in dynamic environments,* Journal of Engineering and Technology Management, v17, p357-377, (2000).

23. Fransman, M., *Information, Knowledge, Vision and Theories of the Firm*, Industrial and Corporate Change v3, n3, p713-757 (1994).

24. Garud, R. and Nayyar, P.R., *Transformative Capacity: Continual Structuring by Intertemporal Technology Transfer,* Strategic Management Journal, v15, p365-385, (1994).

25. Grant, R.M., *Toward a Knowledge-Based Theory of the Firm,* Strategic Management Journal, v17, p109-122, (1996).

26. Graves, S.C., Hausman, W. H. and Schwarz, L. B., *Storage-Retrieval Interleaving in Automatic Warehousing Systems,* Management Science, v23, n9, (1977).

27. Grayling, A.C., *Philosophy 1,* Oxford University Press, (2000).

28. Groover, M. P., *Automation, Production Systems, and Computer-Integrated Manufacturing*, Prentice-Hall Inc., New Jersey, (2001).

29. Gulati, R., *Network Location and Learning: The Influence of Network Resources and Firm Capabilities on Alliance Formation,* Strategic Management Journal, v20, p397-420, (1999).

30. Han, M-H., McGinnis, F.F., Shieh, J.S. and White, A., *On Sequencing Retrievals in an Automated Storage/Retrieval System,* IIE Transactions, v19 n1, p56-66, (1987).

31. Hausman, W.H., Schwarz, L.B. and Graves, S.C., *Optimal Storage Assignment in Automatic Warehousing Systems,* Management Science, v22, n6, (1976).

32. Henderson, R.M. and Clark, K.B., *Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms,* Administrative Science Quarterly, v35, p3-30, (1990).

33. Hitt, M.A., Ireland, R.D. and Hoskisson, R.E., *Strategic Management: Competitiveness and Globalisation,* 3rd Ed., Southwestern College Publishing, (1999).

34. Hitt, M.A., Ireland, R.D. and Lee, H., *Technological Learning, Knowledge Management, Firm Growth and Performance: An Introductory Essay,* Journal of Engineering and Technology Management, v17, n3-4, p231-246, (2000).

35. Hoopes, D.G. and Postrel, S., *Shared Knowledge, "Glitches" and Product Development Performance,* Strategic Management Journal, v20, p837-865, (1999).

36. Itami, H. and Numagami, T., *Dynamic Interaction between Strategy and Technology,* Strategic Management Journal, v13, p119-135, (1992).

37. Kamara, J.M., Anumba, C.J. and Carrillo, P.M., *A CLEVER Approach to Selecting a Knowledge Management Strategy,* International Journal of Project Management, v20, n3, p205-211, (2002).

38. Kersala, A. and Peters, B.A., *An Analysis of Dual Shuttle Automated Storage/Retrieval Systems,* Journal of Manufacturing Systems, (1994).

39. Kogut, B., *Joint Ventures: Theoretical and Empirical Perspectives,* Strategic Management Journal, v9, p319-332, (1988).

40. Kogut, B. and Zander, U., *Knowledge of the Firm, Combinative Capabilities, and the Replication of Technology,* Organisation Science, v3, p383-397, (1992).

41. Lave, J., *The practice of learning: Understanding practice: Perspectives on activity and context,* Cambridge University Press: p3-32, (1993).

42. Lee, H.F. and Schaefer, S.K., *Sequencing Methods for Automated Storage and Retrieval Systems with Dedicated Storage,* Computers and Industrial Engineering, v32, n2, p351-362, (1997).

43. Lei, D., Hitt, M.A. and Bettis, R., *Dynamic Core Competencies Through Meta-Learning and Strategic Context,* Journal of Management, v22, p549-569, (1996).

44. Liebeskind, J.P., Oliver, A.L., Zucker, L. and Brewer, M., *Social Networks, Learning and Flexibility: Sourcing Scientific Knowledge in New Biotechnology Firms,* Organization Science, v7,p428-443, (1996).

45. Maher, T., *Development and Implementation of a Communication System for Distributed Control of a Multi-Machine Manufacturing System,* WRTC, (1997).

46. Mansuri, M., *Cycle-Time Computation and Dedicated Storage Assignment for AS/R Systems,* Computers and Industrial Engineering, v33, n1-2, p307-310, (1997).

47. March, J.G., *Exploration and Exploitation in Organisational Learning,* Organization Science, v2, p71-87, (1991).

48. Marsh, S.J.and Ranft, A.L., *Why Resources Matter: An Empirical Study of the Influence of Knowledge-Based Resources on New Market Entry,* Dynamic Strategic Resources, Wiley, p 43-66, (1999).

49. Matusik, S.F. and Hill, C.W., *The Utilization of Contingent Work, Knowledge Creation and Competitive Advantage,* Academy of Management Review, v23, p680-697, (1998).

50. McKenna, N. and Naughton, P., *Further Development of physical ASRS in WIT*, u/g project report, WIT, (2005).

51. McNelis, G., *A System Modelling Approach to the Development of Flexible Manufacturing and Assembly Cells,* WIT, (2001).

52. Miller, D.A., *A Preliminary Typology of Organisational Learning: Synthesizing the Literature,* Journal of Management, v22, p484-505, (1996).

53. Mitchell M., *The Development of Real-Time Communications and Information Systems for Cells in a Modern Manufacturing Environment,* WIT, (1998).

54. Moon, G. and Kim, G.P., *Effects of Relocation to AS/RS Storage Location Policy with Production Quantity Variation,* Computers and Industrial Engineering, v40, i1-2, p1-13, (2001).

55. Muralidharan, B., Linn, R.J. and Pandit, R., *Shuffling Heuristics for the Storage Location Assignment in an AS/RS,* International Journal in Production Research, v33, n6, p1661-1672, (1995).

56. Nelson, R.R. and Winter, S.G., *An Evolutionary Theory of Economic Change, Belknap Press, (1982).*

57. Nonaka, I., *A Dynamic Theory of Organisational Knowledge Creation,* Organisation Science, v5, p14-37, (1994).

58. O' Connor, A., *The Evolution of a Progressing Interface for a Networked Manufacturing System,* WRTC, (1997).

59. OECD, *The Knowledge-Based Economy,* OCDE/GD(96)102, Paris, (1996)

60. Oliveira, M., *Core Competencies and the Knowledge of the Firm,* Dynamic Strategic Resources, Wiley, p17-41, (1999).

61. O' Mahoney, P., *Automated Storage and Retrieval: System Development and Implementation*, W.I.T. (2004).

62. Phelan, J., *Evolution of a Manufacturing Research Process: experience from an RTC*, IMC-14 conference, TCD, (1997).

63. Phelan, J., and Alexander, R., *Fixture Based Flexible Assembly*, SME Automation and Assembly Conference, Forth Worth, Texas (2004).

64. Polanyi, M., *Personal Knowledge: Towards a Post-Critical Philosophy,* University of Chicago Press, (1958).

65. Polanyi, M., *The Tacit Dimension,* Doubleday Press, (1967).

66. Reed, R. and DeFillippi, *Causal Ambiguity, Barriers to Imitation, and Sustainable Competitive Advantage,* Academy of Management Review, v15, p88-102, (1990).

67. Rembold, U., Nnaji, B. O., and Storr, A., *Computer Integrated Manufacturing and Engineering*, Addison-Wesley, (1993).

68. Sanchez, R. and Heene, A., *A Comparative Perspective on Strategic Learning and Knowledge Management,* Wiley, (1997).

69. Schwarz, L.B., Graves, S.C. and Hausman, W.H., *Scheduling Policies for Automatic Warehousing Systems: Simulation Results,* AIIE Transactions, v10, n3, (1978).

70. Shouldice, C. and Kent, T., *Development of SCADA Control of an ASRS Test-Bed,* u/g project report, WIT, (2006).

71. Simonin, B.L., *Ambiguity and the Process of Knowledge Transfer in Strategic Alliances,* Strategic Management Journal, v20, p595-623, (1999).

72. Singh, K., *The Impact of Technological Complexity and Interfirm Collaboration on Business Survival,* Academy of Management Journal, v40, p339-367, (1997).

73. Spender, J.C., *Making Knowledge the Basis of a Dynamic Theory of the Firm,* Strategic Management Journal, v17, p45-62, (1996).

74. Spender, J.C. and Grant, R.M., *Knowledge and the Firm: Overview,* Strategic Management Journal, v17, p5-9, (1996).

75. Subramaniam, M. and Venkatraman, N., *The influence of Leveraging Tacit Overseas Knowledge for Global New Product Development Capability: An Empirical Examination,* Dynamic Strategic Resources, Wiley, (1999).

76. Taboun, S.M. and Bhole, S.D., *A Simulator for an Automated Warehousing System,* Computers and Industrial Engineering, v24, n2, p281-290, (1993).

77. Teece, D.J., Pisano, G. and Shuen, A., *Dynamic Capabilities and Strategic Management,* Strategic management Journal, v18, p509-533, (1997).

78. Trought, B., *The Knowledge Question for Manufacturing,* Management Research International, (2001).

79. Van Den Berg, J.P., and Gademann, A.J.R.M., *Simulation Study of an Automated Storage/Retrieval System,* International Journal of Production Research, v38, n6, p1339-1356, (2000).

80. Walsh, D., *An Automated Storage / Retrieval System Simulation Model, IMC 21, (2004).*

81. White, J.A., and Kinney, H.D., *Storage and Warehousing,* Handbook of Industrial Engineering, John Wiley and Sons Inc., (1982).

82. Zahra, S.A., Melsen, A.P. and Bogner, W.C., *Corporate Entrepreneurship, Knowledge and Competence Development,* Entrepreneurship: Theory and Practice, v23, n3, p169-189, (1999).

Bibliography

1. Siemens AG, *SITRAIN Training for Automation and Drives*, Revision A, (2002).
2. Siemens AG, *Simatic HMI, WinCC V5 Getting Started*, (2003).
3. Minarik Automation and Control, *Adjustable Speed Drives for DC Motors,* (1997)
4. Siemens AG, *SIMATIC WinCC Add-on, DBExt DLL,* (2002)

Glossary

2D	Two Dimensional
3D	Three Dimensional
AGVS	Automatic Guided Vehicle System
AMT	Advanced manufacturing Technology
ASRS	Automated Storage and Retrieval System
CMM	Coordinate Measuring Machine
CNC	Computer Numerical Control
DC	Dual Cycle
DDE	Dynamic Data Exchange
DLL	Dynamic Link Library
ERP	Enterprise Resource Management
FCFS	First Come First Served
FIFO	First In First Out
FMAS	Flexible Manufacturing and Flexible Assembly System
FMC	Flexible Manufacturing Cell
FMS	Flexible Manufacturing System
HMI	Human Machine Interface
I/O	Input/Output
KM	Knowledge Management
MCL	Most Common Locations
MIL	Mandatory Interleaving
MTCR	Modified Traffic Congestion Ratio
NIL	No Interleaving
ODBC	Open Data Base Connectivity
OECD	Organisation for Economic Co-Operation and Development
OPC	Open Connectivity
OLE	Object Linking and Embedding
PC	Personal Computer
P&D	Pickup and Delivery
PLC	Programmable Logic Controller
R&D	Research and Development
RAN	Random Storage Assignment

RCD	Residual Current Device
RPM	Revolutions per Minute
SC	Single Cycle
SCADA	Supervisory Control and Data Acquisition
SCARA	Selective Compliance Assembly Robot Arm
SECI	Socialisation, Externalisation, Combination, Internalisation
SKU	Stock Keeping Unit
SPT	Shortest Processing Time
SQL	Structured Query Language
S/R	Storage / Retrieval
WIT	Waterford Institute of Technology

Appendix A (PLC Program)

Please consult author for complete PLC program listings.

Appendix B (SCADA Screens)

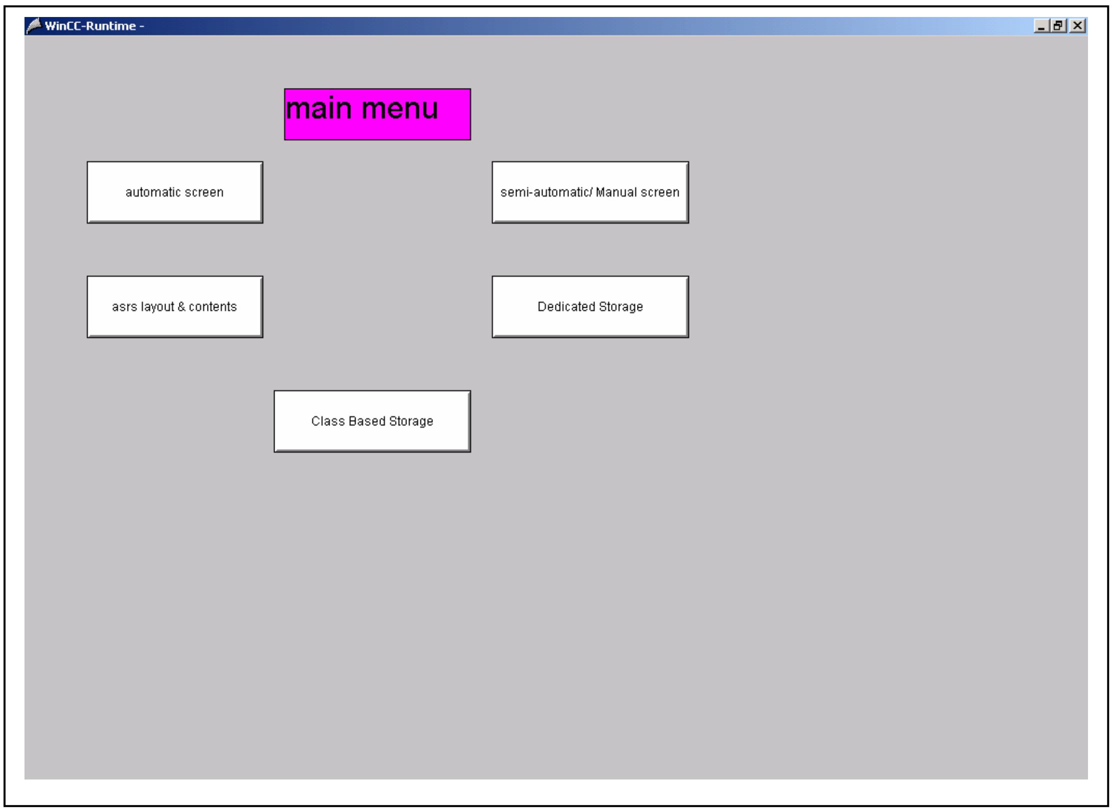


Figure B-1: Main Menu

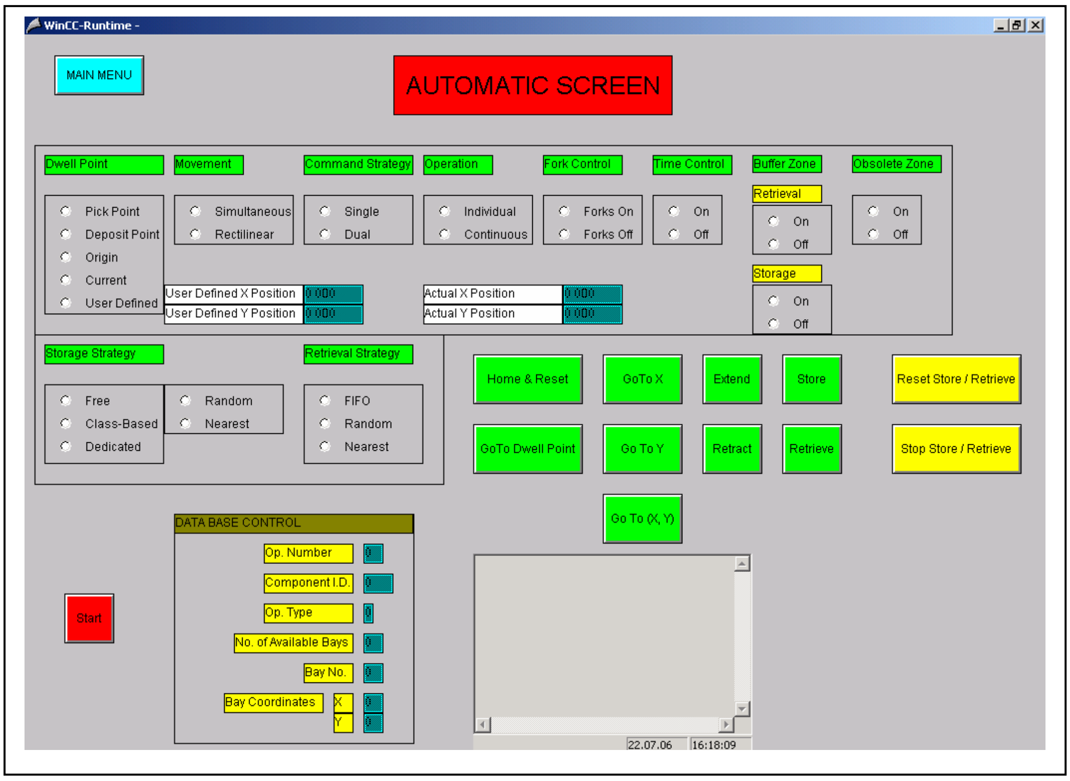


Figure B-2: Semi / Full Automatic Screen

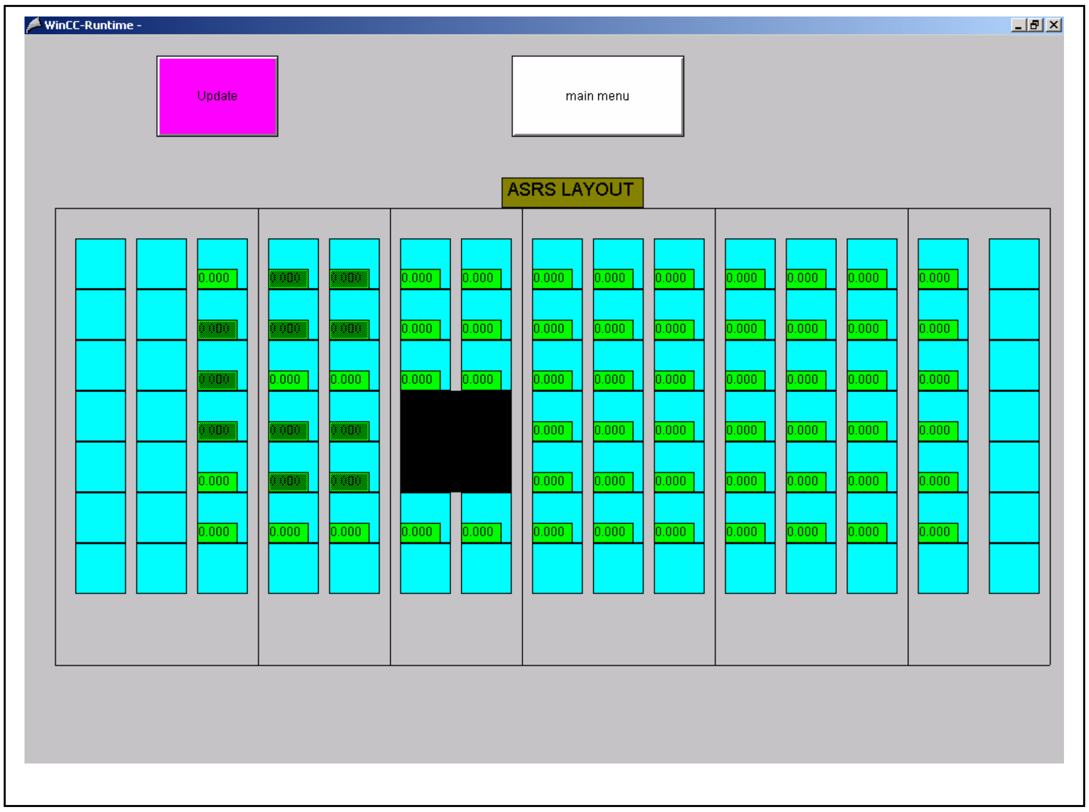


Figure B-3: ASRS Layout and Contents

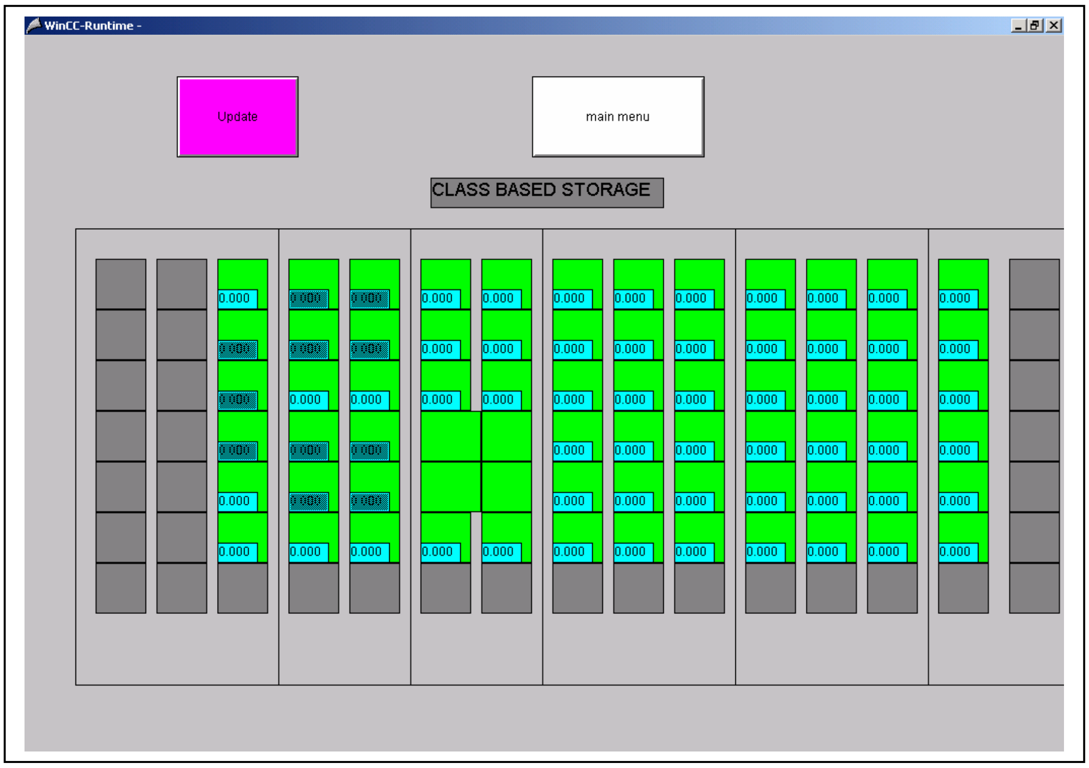


Figure B-4: Class-Based Storage

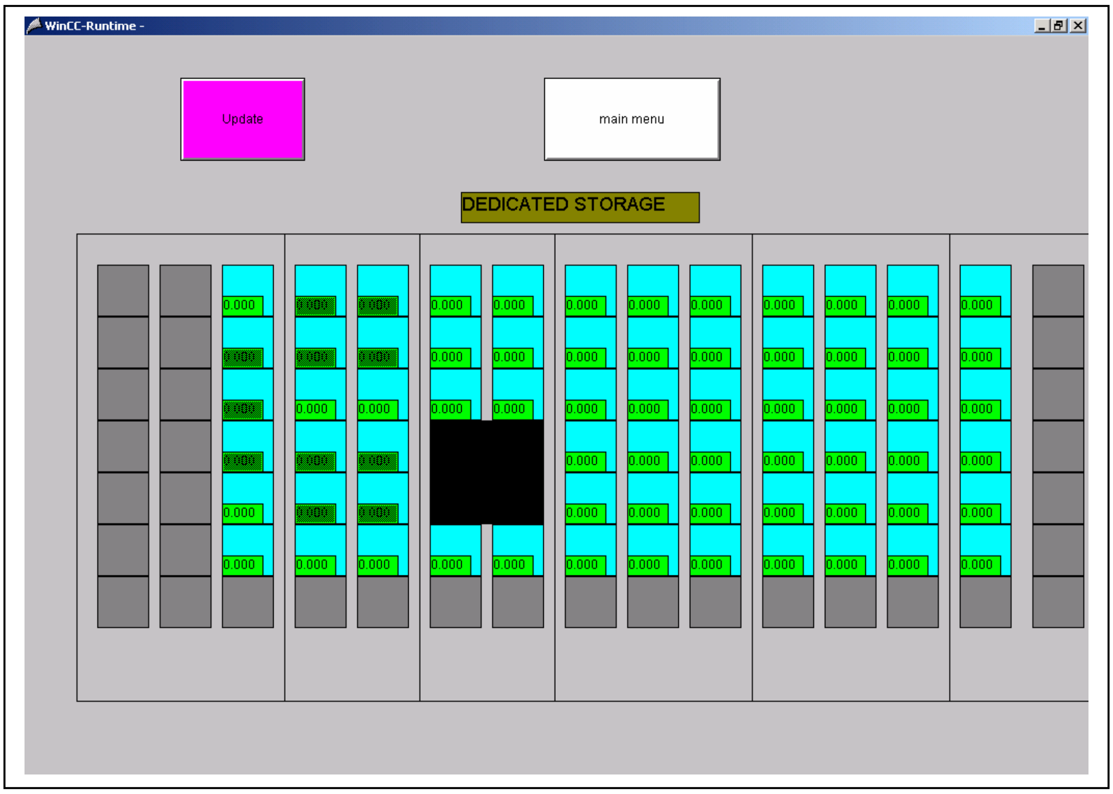


Figure B-5: Dedicated Storage

Appendix C (SCADA Scripts)

ASRSOp

```
#include "apdefap.h"

long BayNo, OpNumber, OpNumber1, OpReqd, PartNo, OpType=0, NoResult =0;
long BayNoRequired, CompID, AutoID, AutoID1, ClassID; //CompID = DB Update, AutoID = DB
Search
long XAct, YAct, XStartPosn, YStartPosn;
long BayArray[68] = {0};
//long lUserData = 1234;
long OpNo = 1, Checked;
long OpComp;
CMN_ERROR MyError;
char*  PictureName;
char MyDataSource[] = "ASRSDatabase";
char sqlCond[100]  ;
char sqlCond1[100];
char sqlCond2[100];
char sqlCond3[100];
char strIOField[11] ;
//char strIOField1[] ="G";
char strSQLHoldComp[]=" ComponentID = ";
char strSQLDedicated[]=" Dedicated = ";
char strSQLHoldBay[]=" BayNumber = ";
char strSQLClass[]=" Class = ";
char strSQLORDERBYDate[]=" ORDER BY DateIn ASC";
char strSQLOpReqd[]=" (OperationReqd = 1 OR OperationReqd = 2) AND Checked = 0";
char strSQLOpNumber[]=" OpNumber= ";
char strSQLNULL[]="\0";
char strSQLOperationReqd[]=" OperationReqd= ";
char strSQLPartNumber[]=" PartNumber = ";
char TableName[] = "ASRSContents";
char TableName1[]="ClassType";
char TableName2[]="Requirements";
char TableName3[]="OperationsCompleted";
char strFieldHold[]="IOField";
char strMsg[20] = "All OK";
char strMsg1[20] = "Finished";
//char strSQLHold2[]=" XCoOrdinate = ";
//char strSQLHold3[]=" YCoOrdinate = ";
int  r=0, i=0, j, XCo, YCo, bCheck = 0, XPick = 4, YPick = 3, XDeposit = 5, YDeposit = 3;
//WORD T1=13, T2=23;
DWORD  wRecs;
struct tm *OldT;
time_t LiamTime;
double Dist = 0, Disti =1000,Atime = 0.0, TimeDifference = 0.0, DateTimeReqd, DateTimeComp;
BOOL Result ;



DBEXT_VARIABLE_STRUCT CoOrdVar [] = {
{"BayNumber",&BayNo,DBE_VS_TYP_VARIABLE},
{"XCoOrdinate",&XCo,DBE_VS_TYP_VARIABLE},
{"YCoOrdinate",&YCo,DBE_VS_TYP_VARIABLE},
{"ComponentID",&AutoID,DBE_VS_TYP_VARIABLE},
{NULL,NULL,0}
};


DBEXT_VARIABLE_STRUCT CoOrdVarPut [] = {
```

```
{"ComponentID",&CompID,DBE_VS_TYP_VARIABLE},
{"DateIn",&Atime,DBE_VS_TYP_VARIABLE},
{NULL,NULL,0}
};

DBEXT_VARIABLE_STRUCT CheckedPut [] = {
{"Checked",&bCheck,DBE_VS_TYP_VARIABLE},
{NULL,NULL,0}
};

DBEXT_VARIABLE_STRUCT BayPut [] = {
{"BayNumber",&BayNo,DBE_VS_TYP_VARIABLE},
{NULL,NULL,0}
};

DBEXT_VARIABLE_STRUCT Classified [] = {
{"Class",&ClassID,DBE_VS_TYP_VARIABLE},
{NULL,NULL,0}
};

DBEXT_VARIABLE_STRUCT Requirements [] = {
{"OpNumber",&OpNumber,DBE_VS_TYP_VARIABLE},
{"PartNumber",&PartNo,DBE_VS_TYP_VARIABLE},
{"OperationReqd",&OpReqd,DBE_VS_TYP_VARIABLE},
{"DateTimeReqd",&DateTimeReqd,DBE_VS_TYP_VARIABLE},
{"Checked",&Checked,DBE_VS_TYP_VARIABLE},
{NULL,NULL,0}
};

DBEXT_VARIABLE_STRUCT OperationsCompleted [] = {
{"OperationNumber",&OpNumber1,DBE_VS_TYP_VARIABLE},
{"PartNumber",&PartNo,DBE_VS_TYP_VARIABLE},
{"OperationComp",&OpComp,DBE_VS_TYP_VARIABLE},
{"DateTime",&DateTimeComp,DBE_VS_TYP_VARIABLE},
{NULL,NULL,0}
};

void ASRSOp()
{
printf("Start ASRSOp \n");


do
{
//printf("Start Do \n");
//SetTagBit("LatchGoToXYSimRect",0);
SetTagBit("TimeDelay",0);

if (OpType !=0)
{
sprintf(sqlCond,"OperationReqd = %d", OpType);
strcat(sqlCond," AND Checked = 0");
}// end if

else
{
sprintf(sqlCond,"%s", strSQLOpReqd);
}//end else
//strcat(sqlCond," AND Checked = 0");
```

```
//printf("OpType = %d \n", OpType);
Result =
DBEGetVariables(PictureName,MyDataSource,TableName2,sqlCond,Requirements,&MyError);
if (!Result)
{
if ((OpType !=0) && (NoResult == 0))
{
OpType = 0;
NoResult = 1;
SetTagBit("ASRSOpOpTypeNotAvailable",1);
//printf("!Result & OpType != 0 & NoResult ==0 \n");
//OpNumber = 998;
//ASRSDwell();
goto end;

}// end if OpType & NoResult
printf("No Op.  Available - Requirements Need to be Modified \n");
SetTagBit("ASRSOpNoRequirements",0);
goto end;
}// end if !Result
//printf("PartNumber = %ld\n", PartNo);
//printf("OperationReqd = %ld\n", OpReqd);

OpComp = OpReqd;
//printf("OperationNumber = %ld\n", OpNumber);
//printf("PartNumber = %ld\n", PartNo);
//printf("OpComp = %ld\n", OpComp);
//printf("DateTimeComp = %ld\n", DateTimeComp);

//goto finish;
//printf("Test \n");

LiamTime=time(NULL);
OldT = localtime(&LiamTime);
Atime = (LiamTime/3600.0/24.0)+25569.0;
if (OldT->tm_isdst ==1)
      Atime = Atime + 1.0/24.0;
//printf("Time is %ld\n",LiamTime);
SetTagWord("Liam1",0);
SetTagWord("Liam2",0);
SetTagWord("autocomponentID",0);
SetTagWord("XCoOrdinateDB",0);
SetTagWord("YCoOrdinateDB",0);
SetTagWord("OpReqdOP",0);
SetTagWord("OpNumberOP",0);

DateTimeComp = Atime;

AutoID1=PartNo;


if (GetTagBit("TimeControl") ==1)
{
//printf("DateTimeReqd = %lf\n", DateTimeReqd);
//printf("Atime = %lf\n", Atime);
TimeDifference = ((DateTimeReqd - Atime)*86400.0);

//printf("TimeDifference = %lf seconds \n", TimeDifference);
if (TimeDifference > 5)
{
```

```
SetTagBit("TimeDelay",1);
printf("5 Sec. Time Delay Operational \n");
goto end;

}

}// end if TimeControl


if (OpReqd == 1)
{
CompID=PartNo;
AutoID = 0;
XStartPosn = XPick;
YStartPosn = YPick;
} // end if Store


if  (OpReqd == 2)
{
AutoID =PartNo;
CompID = 0;
XStartPosn = GetTagWord("xAUTOActPos");
YStartPosn = GetTagWord ("yAUTOActPos");
}// end if Retrieve


if ((GetTagWord("Random") ==1) && (OpReqd == 1)&&(GetTagWord("Free")==1))
{
printf("Random Free Storage selected. \n");
goto random;

}//end if Random Free Storage


if ((GetTagWord("Free") ==1) && (OpReqd == 1) && (GetTagWord("nearest") ==1) )
{
printf("Nearest Free Storage selected. \n");
goto nearest;

}//end if Nearest Free Storage


if ((GetTagWord("Classbased") ==1) && (OpReqd == 1) && (GetTagWord("Random") ==1) )
{
printf("Class-Based Random Storage selected. \n");
goto classbased;

}//end if Class-Based Random Storage


if ((GetTagWord("Classbased") ==1) && (OpReqd == 1) && (GetTagWord("nearest") ==1) )
{
printf("Class-Based Nearest Storage selected. \n");
goto classbased;

}//end if Class-Based Nearest Storage


if ((GetTagWord("Dedicated") ==1) && (OpReqd == 1) && (GetTagWord("Random") ==1) )
```

```
{
printf("Dedicated Random Storage selected. \n");
goto dedicated;

}//end if Dedicated Random Storage


if ((GetTagWord("Dedicated") ==1) && (OpReqd == 1) && (GetTagWord("nearest") ==1) )
{
printf("Dedicated Nearest Storage selected. \n");
goto dedicated;

}//end if Dedicated Nearest Storage





if ((GetTagWord("FIFO") ==1) && (OpReqd == 2))
{
printf("FIFO Retrieval selected. \n");
goto fifo;

}//end if FIFO Retrieval



if ((GetTagWord("Random_1") ==1) && (OpReqd == 2))
{
printf("Random Retrieval selected. \n");
goto random;

}//end if Random Retrieval



if ((GetTagWord("nearest_1") ==1) && (OpReqd == 2))
{
printf("Nearest Retrieval selected. \n");
goto nearest;

}//end if Nearest Retrieval


strcpy (strMsg, "Storage / Retrieval strategy not selected");

goto normal;




random:
//ASRSRandom;
sprintf(sqlCond,"%s%d",strSQLHoldComp,AutoID);
Result = DBEGetNumberOfRecords(MyDataSource,TableName,sqlCond,&wRecs,&MyError);

switch (wRecs)
{
case 0 :
            strcpy (strMsg, "None Available");
```

```
if (OpReqd == 1)
        sprintf(sqlCond,"%s%d",strSQLOperationReqd,OpReqd);
if (OpReqd == 2)
        sprintf(sqlCond,"%s%dAND%s%d",strSQLOperationReqd,OpReqd,strSQLPartNumber,PartNo);

goto error1;

case 1 :

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
                break;
default :
strcat (sqlCond , " AND Checked = 0");
bCheck = 1;

for (r = 0; r < wRecs ; r++)
{

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
sprintf( sqlCond1, "%s%d", strSQLHoldBay, BayNo);
Result=DBESetVariables(PictureName,MyDataSource,TableName,sqlCond1,CheckedPut,&MyError);

BayArray[r] = BayNo;
}//end for r
strcpy(sqlCond, " Checked = 1 ");
bCheck = 0;
Result=DBESetVariables(PictureName,MyDataSource,TableName,sqlCond,CheckedPut,&MyError);
srand(time(NULL));
r=rand( )%wRecs; //random number  0 to No. of data items in the array
sprintf(sqlCond,"%s%d",strSQLHoldBay, BayArray[r]);

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
if   (!Result)
{
strcpy (strMsg, "Can't Get Co=ordinates");
goto error;
}
} // end switch wRecs

goto normal;

nearest:

sprintf(sqlCond,"%s%d",strSQLHoldComp,AutoID);
//printf("sqlCond = %s\n",sqlCond);
Result = DBEGetNumberOfRecords(MyDataSource,TableName,sqlCond,&wRecs,&MyError);

switch (wRecs)
{
case 0 :
               strcpy (strMsg, "None Available");
if (OpReqd == 1)
        sprintf(sqlCond,"%s%d",strSQLOperationReqd,OpReqd);
if (OpReqd == 2)
        sprintf(sqlCond,"%s%dAND%s%d",strSQLOperationReqd,OpReqd,strSQLPartNumber,PartNo);
               goto error1;
case 1 :

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
                break;
```

```
default :
sprintf(sqlCond ,"%s%d AND Checked = 0",strSQLHoldComp, AutoID);
bCheck = 1;
//printf("sqlCond = %s\n",sqlCond);
for (r = 0; r < wRecs ; r++)
{

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
sprintf( sqlCond1, "%s%d", strSQLHoldBay, BayNo);
Result=DBESetVariables(PictureName,MyDataSource,TableName,sqlCond1,CheckedPut,&MyError);
BayArray[r] = 0;
BayArray[r] = BayNo;
}//end for r
strcpy(sqlCond, " Checked = 1 ");
bCheck = 0;
Result=DBESetVariables(PictureName,MyDataSource,TableName,sqlCond,CheckedPut,&MyError);
Disti=1000;
for (i = 0; i < wRecs ; i++)
{
//printf("wRecs= %d\n", wRecs);

sprintf(sqlCond,"%s%d", strSQLHoldBay, BayArray[i]);
Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
//printf("BayNumber = %d\n", BayArray[i]);
Dist = sqrt(((XStartPosn - XCo)*(XStartPosn - XCo)) + ((YStartPosn - YCo)*(YStartPosn - YCo)));
//printf("Distance = %lf\n", Dist);
//printf("Distancei = %lf\n", Disti);

if (Dist < Disti)
{
Disti = Dist;
BayNoRequired = 0;
BayNoRequired = BayArray[i];
}//end if


}//end for i

//printf("BayNoRequired = %d\n", BayNoRequired);

sprintf(sqlCond,"%s%d",strSQLHoldBay, BayNoRequired);
//printf("sqlCond = %s\n",sqlCond);
Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
if   (!Result)
     {strcpy (strMsg, "Can't Get Co=ordinates");
     goto error;
}
} // end switch wRecs

goto normal;


dedicated:

sprintf(sqlCond,"%s%d",strSQLHoldComp,AutoID);
sprintf(sqlCond2,"%s%d AND%s%d",strSQLHoldComp,AutoID,strSQLDedicated,CompID);
Result = DBEGetNumberOfRecords(MyDataSource,TableName,sqlCond2,&wRecs,&MyError);

//printf("%s \n",sqlCond2);
//printf("\n");
```

```
switch (wRecs)
{
case 0 :
            strcpy (strMsg, "None Available");

sprintf(sqlCond,"%s%dAND%s%d",strSQLOperationReqd,OpReqd,strSQLPartNumber,PartNo);
            goto error1;

case 1 :

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond2,CoOrdVar,&MyError);
             strcpy(sqlCond,sqlCond2);
             break;
default :
strcat (sqlCond2 , " AND Checked = 0");
bCheck = 1;

for (i = 0; i < wRecs ; i++)
{

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond2,CoOrdVar,&MyError);
sprintf( sqlCond1, "%s%d", strSQLHoldBay, BayNo);
Result=DBESetVariables(PictureName,MyDataSource,TableName,sqlCond1,CheckedPut,&MyError);

BayArray[i] = BayNo;
}//end for i
strcpy(sqlCond, " Checked = 1 ");
bCheck = 0;
Result=DBESetVariables(PictureName,MyDataSource,TableName,sqlCond,CheckedPut,&MyError);
//exit(0);

if (GetTagWord("Random")==1)
{
srand(time(NULL));
r=rand( )%wRecs; //random number  0 to No. of data items in the array
sprintf(sqlCond,"%s%d",strSQLHoldBay, BayArray[r]);
}//end if

else
{
Disti=1000;
for (i = 0; i < wRecs ; i++)
{

sprintf(sqlCond,"%s%d", strSQLHoldBay, BayArray[i]);
Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);

Dist = sqrt(((XStartPosn - XCo)*(XStartPosn - XCo)) + ((YStartPosn - YCo)*(YStartPosn - YCo)));

if (Dist < Disti)
{Disti = Dist;
BayNoRequired = BayArray[i];

}//end if
}//end for i
//printf("BayNoRequired = %d\n", BayNoRequired);
sprintf(sqlCond,"%s%d",strSQLHoldBay, BayNoRequired);

}//end else
```

```
Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
if  (!Result)
    {strcpy (strMsg, "Can't Get Co=ordinates");
    goto error;
}// end if

} // end switch wRecs

goto normal;


classbased:

sprintf(sqlCond,"%s%d",strSQLHoldComp,CompID);
//printf("%s \n",sqlCond);
//printf("\n");
Result=DBEGetVariables(PictureName,MyDataSource,TableName1,sqlCond,Classified,&MyError);

sprintf(sqlCond,"%s%d",strSQLHoldComp,AutoID);
sprintf(sqlCond2,"%s%d AND%s%d",strSQLHoldComp,AutoID,strSQLClass,ClassID);
Result = DBEGetNumberOfRecords(MyDataSource,TableName,sqlCond2,&wRecs,&MyError);

//printf("%s \n",sqlCond2);
//printf("\n");

switch (wRecs)
{
case 0 :
             strcpy (strMsg, "None Available");
sprintf(sqlCond,"%s%dAND%s%d",strSQLOperationReqd,OpReqd,strSQLPartNumber,PartNo);
            goto error1;

case 1 :

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond2,CoOrdVar,&MyError);
              strcpy(sqlCond,sqlCond2);
              break;
default :
strcat (sqlCond2 , " AND Checked = 0");
bCheck = 1;

for (r = 0; r < wRecs ; r++)
{

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond2,CoOrdVar,&MyError);
sprintf( sqlCond1, "%s%d", strSQLHoldBay, BayNo);
Result=DBESetVariables(PictureName,MyDataSource,TableName,sqlCond1,CheckedPut,&MyError);

BayArray[r] = BayNo;
}//end for r
strcpy(sqlCond, " Checked = 1 ");
bCheck = 0;
Result=DBESetVariables(PictureName,MyDataSource,TableName,sqlCond,CheckedPut,&MyError);
if  (!Result)
{strcpy (strMsg, "Can't Update Checked");
goto error;
}

if (GetTagWord("Random")==1)
{
```

```
srand(time(NULL));
r=rand( )%wRecs; //random number  0 to No. of data items in the array
sprintf(sqlCond,"%s%d",strSQLHoldBay, BayArray[r]);
}//end if

else
{
Disti=1000;
for (i = 0; i < wRecs ; i++)
{

sprintf(sqlCond,"%s%d", strSQLHoldBay, BayArray[i]);
Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);

Dist = sqrt(((XStartPosn - XCo)*(XStartPosn - XCo)) + ((YStartPosn - YCo)*(YStartPosn - YCo)));

if (Dist < Disti)
{Disti = Dist;
BayNoRequired = BayArray[i];

}//end if
}//end for i
//printf("BayNoRequired = %d\n", BayNoRequired);
sprintf(sqlCond,"%s%d",strSQLHoldBay, BayNoRequired);

}//end else
Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
if   (!Result)
    {strcpy (strMsg, "Can't Get Co=ordinates");
    goto error;
}// end if


} // end switch wRecs

goto normal;




fifo:

sprintf(sqlCond,"%s%d%s",strSQLHoldComp,AutoID,strSQLORDERBYDate);
//printf("%s \n",sqlCond);
//printf("\n");

Result=DBEGetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVar,&MyError);
if (!Result)
{               strcpy (strMsg, "None Available");
sprintf(sqlCond,"%s%dAND%s%d",strSQLOperationReqd,OpReqd,strSQLPartNumber,PartNo);
                goto error1;
}
sprintf(sqlCond,"%s%d",strSQLHoldBay,BayNo);
//printf("%s \n",sqlCond);
//printf("\n");

goto normal;

error1:
```

```
bCheck = 1;
Result=DBESetVariables(PictureName,MyDataSource,TableName2,sqlCond,CheckedPut,&MyError);




goto finish;

error:
printf("\n%s", strMsg);
goto finish;

normal:

//printf("%s\n", strMsg);
SetTagWord("Liam1",wRecs);
SetTagWord("Liam2",BayNo);
SetTagWord("autocomponentID",AutoID1);
SetTagWord("XCoOrdinateDB",XCo);
SetTagWord("YCoOrdinateDB",YCo);
SetTagWord("OpReqdOP", OpReqd);
SetTagWord("OpNumberOP", OpNumber);

if (GetTagBit("ASRSOpCompDual")==1)
{
ASRSOpAction();
}


if (GetTagBit("ASRSDwellContinuous")==1)
{
SetTagBit("PickReqd",1);
SetTagBit("ASRSDwellContinuous",0);
SetTagBit("ASRSOpContinuous",1);
if(GetTagBit("ASRSDwellAtXYPosition")==1)
{
SetTagBit("ASRSDwellAtXYPosition",0);
SetTagBit("AtXYPosition",1);
}
}



//printf("Going to GoToXY in ASRSOp \n");

if (GetTagBit("Start")==1)
{
ASRSOpAction();
}


//SetTagBit("go3",1);
//SetTagBit("callauto",1);
//SetTagBit("callsemiauto",0);
//sprintf(sqlCond3,"%s",strSQLNULL);
//printf("sqlCond3 = %s\n", sqlCond3);
OpNumber1=OpNumber;
OpNumber = 999;
NoResult = 0;
//printf("Press Enter to Continue");
```

```
//fscanf(stdin,"%c");

// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END
finish:


printf("%s\n", strMsg1);
//j++;
//printf("OpNumber = %ld\n", OpNumber);
//printf("Job Number %d complete \n", j);
//printf("End Do \n");
}//end do
while (OpNumber != 999);
printf("End Do-While\n");

end:
printf("End \n");
SetTagBit("GoToX",0);
SetTagBit("GoToY",0);

printf("End ASRSOp \n");

}
```

ASRSOpAction

```
void ASRSOpAction()
{
extern long BayNo, OpNumber, OpNumber1, OpReqd, PartNo, OpType, AutoID;
extern long BayNoRequired, CompID, AutoID1, ClassID; //CompID = DB Update, AutoID = DB
Search
extern long XAct, YAct, XStartPosn, YStartPosn;
//extern long BayArray[68];
extern long OpNo, DateTimeReqd, Checked;
extern long OpComp, DateTimeComp;
extern CMN_ERROR MyError;
extern char*  PictureName;
extern char MyDataSource[] ;
extern char sqlCond[100]  ;
extern char sqlCond1[100];
extern char sqlCond2[100];
extern char sqlCond3[100];
extern char strIOField[11] ;
extern char strSQLHoldComp[];
extern char strSQLDedicated[];
extern char strSQLHoldBay[];
extern char strSQLClass[];
extern char strSQLORDERBYDate[];
extern char strSQLOpReqd[];
extern char strSQLOpNumber[];
extern char strSQLNULL[];
extern char strSQLOperationReqd[];
extern char strSQLPartNumber[];
extern char TableName[];
extern char TableName1[];
extern char TableName2[];
extern char TableName3[];
extern char strFieldHold[];
extern char strMsg[20];
extern char strMsg1[20];
extern int  r, i, j, bCheck, XPick, YPick, XDeposit, YDeposit;
extern int XCo, YCo;
extern DWORD  wRecs;
extern struct tm *OldT;
extern time_t LiamTime;
extern double Dist, Disti,Atime;
extern BOOL Result ;

printf("Start ASRSOpAction \n");

if (OpReqd == 1)
{
SetTagWord("xAUTOReqPos",XPick);
SetTagWord("yAUTOReqPos",YPick);
}//end if

if (OpReqd == 2)
{
SetTagWord("xAUTOReqPos",XCo);
SetTagWord("yAUTOReqPos",YCo);
}//end if
SetTagBit("PickReqd",1);
if (GetTagBit("ASRSOpCompDual")==1)
{
SetTagBit("ASRSOpCompDual",0);
SetTagBit("ASRSOpDual",1);
}


if (GetTagBit("ASRSDwellContinuous")==1)
{
SetTagBit("ASRSDwellContinuous",0);
SetTagBit("ASRSOpContinuous",1);
}



//printf("Going to GoToXY in ASRSOp \n");

if (GetTagBit("Start")==1)
{
SetTagBit("Start",0);
SetTagBit("ASRSOpStart",1);

if((GetTagBit("DwellPickPoint")==1)&&(OpReqd == 1) && (GetTagWord("xAUTOActPos")==4)
&& (GetTagWord("yAUTOActPos")==3))
{
SetTagBit("AtXYPosition",1);
}//end if
else
if((GetTagBit("CurrentLocation")==1)&&(OpReqd == 2) &&
(GetTagWord("xAUTOActPos")==XCo) && (GetTagWord("yAUTOActPos")==YCo))
{
SetTagBit("AtXYPosition",1);
}//end if
else
if((GetTagBit("UserDefined")==1)&&(OpReqd == 2) && (GetTagWord("xAUTOActPos")==XCo)
&& (GetTagWord("yAUTOActPos")==YCo))
{
SetTagBit("AtXYPosition",1);
}//end if



else
{
SetTagBit("GoToXY",1);
}
}
//printf("Here I am");


printf("End ASRSOpAction \n");
}
```

SetPlaceCoOrds

```
#include "apdefap.h"

extern int XPick, YPick, XDeposit, YDeposit, XCo, YCo;
extern long OpReqd;

void SetPlaceCoOrds()
{
printf("Start SetPlaceCoOrds \n");

if (OpReqd == 1)
{
SetTagWord("xAUTOReqPos",XCo);
SetTagWord("yAUTOReqPos",YCo);
}//end if

if (OpReqd == 2)
{
SetTagWord("xAUTOReqPos",XDeposit);
SetTagWord("yAUTOReqPos",YDeposit);
}//end if
//SetTagBit("PlaceReqd",1);
SetTagBit("SetPlaceCoOrdsPlaceReqd",1);
SetTagBit("GoToXY",1);

printf("End SetPlaceCoOrds \n");

}
```

ASRSOpComp

```
#include "apdefap.h"

extern long BayNo, OpNumber, OpNumber1, OpReqd, PartNo, OpType, AutoID;
extern long BayNoRequired, CompID, AutoID1, ClassID; //CompID = DB Update, AutoID = DB
Search
extern long XAct, YAct, XStartPosn, YStartPosn;
//extern long BayArray[68];
extern long OpNo, DateTimeReqd, Checked;
extern long OpComp, DateTimeComp;
extern CMN_ERROR MyError;
extern char*  PictureName;
extern char MyDataSource[] ;
extern char sqlCond[100]  ;
extern char sqlCond1[100];
extern char sqlCond2[100];
extern char sqlCond3[100];
extern char strIOField[11] ;
extern char strSQLHoldComp[];
extern char strSQLDedicated[];
extern char strSQLHoldBay[];
extern char strSQLClass[];
extern char strSQLORDERBYDate[];
extern char strSQLOpReqd[];
extern char strSQLOpNumber[];
extern char strSQLNULL[];
extern char strSQLOperationReqd[];
extern char strSQLPartNumber[];
extern char TableName[];
extern char TableName1[];
extern char TableName2[];
extern char TableName3[];
extern char strFieldHold[];
extern char strMsg[20];
extern char strMsg1[20];
extern int  r, i, j, bCheck, XPick, YPick, XDeposit, YDeposit;
extern int XCo, YCo;
extern DWORD  wRecs;
extern struct tm *OldT;
extern time_t LiamTime;
extern double Dist, Disti,Atime;
extern BOOL Result ;

//extern DBEXT_VARIABLE_STRUCT CoOrdVar []= {
//{"BayNumber",&BayNo,DBE_VS_TYP_VARIABLE},
//{"XCoOrdinate",&XCo,DBE_VS_TYP_VARIABLE},
//{"YCoOrdinate",&YCo,DBE_VS_TYP_VARIABLE},
//{"ComponentID",&AutoID,DBE_VS_TYP_VARIABLE},
//{NULL,NULL,0}
//};

extern DBEXT_VARIABLE_STRUCT CoOrdVarPut [];// = {
//{"ComponentID",&CompID,DBE_VS_TYP_VARIABLE},
//{"DateIn",&Atime,DBE_VS_TYP_VARIABLE},
//{NULL,NULL,0}
//};

extern DBEXT_VARIABLE_STRUCT CheckedPut [];// = {
//{"Checked",&bCheck,DBE_VS_TYP_VARIABLE},
//{NULL,NULL,0}
//};

//extern DBEXT_VARIABLE_STRUCT BayPut [] = {
//{"BayNumber",&BayNo,DBE_VS_TYP_VARIABLE},
//{NULL,NULL,0}
//};

//extern DBEXT_VARIABLE_STRUCT Classified [] = {
//{"Class",&ClassID,DBE_VS_TYP_VARIABLE},
//{NULL,NULL,0}
//};

//extern DBEXT_VARIABLE_STRUCT Requirements [] = {
//{"OpNumber",&OpNumber,DBE_VS_TYP_VARIABLE},
//{"PartNumber",&PartNo,DBE_VS_TYP_VARIABLE},
//{"OperationReqd",&OpReqd,DBE_VS_TYP_VARIABLE},
//{"DateTimeReqd",&DateTimeReqd,DBE_VS_TYP_VARIABLE},
//{"Checked",&Checked,DBE_VS_TYP_VARIABLE},
//{NULL,NULL,0}
//};

extern DBEXT_VARIABLE_STRUCT OperationsCompleted [];// = {
//{"OperationNumber",&OpNumber,DBE_VS_TYP_VARIABLE},
//{"PartNumber",&PartNo,DBE_VS_TYP_VARIABLE},
//{"OperationComp",&OpComp,DBE_VS_TYP_VARIABLE},
//{"DateTime",&DateTimeComp,DBE_VS_TYP_VARIABLE},
//{NULL,NULL,0}
//};

void ASRSOpComp()
{



printf("Start ASRSOpComp \n");
Result=DBESetVariables(PictureName,MyDataSource,TableName,sqlCond,CoOrdVarPut,&MyError);
if (!Result)
{
strcpy (strMsg, "Did not update");

//goto error;
}// end if (!Result)
else
{
strcpy (strMsg, "All OK");
}

strcpy(sqlCond, " Checked = 1 ");
bCheck = 0;
Result=DBESetVariables(PictureName,MyDataSource,TableName2,sqlCond,CheckedPut,&MyError);

sprintf(sqlCond1,"%s%d",strSQLOpNumber,OpNumber1);
Result = DBEDeleteRecord(MyDataSource,TableName2,sqlCond1,&MyError);

Result =
DBESetVariables(PictureName,MyDataSource,TableName3,NULL,OperationsCompleted,&MyError);

strcpy(sqlCond, " Checked = 1 ");
bCheck = 0;
```

```
Result=DBESetVariables(PictureName,MyDataSource,TableName2,sqlCond,CheckedPut,&MyError);

if (GetTagBit("Single")==1)
{
  printf("Single Cycle Selected \n");
  OpType=0;
SetTagBit("ASRSOpCompSingle",1);
}//end if Single

if (GetTagBit("Dual")==1)
{
printf("Dual Cycle Selected \n");
if ((OpType==1)||(OpType==2))
{
OpType=0;
SetTagBit("ASRSOpCompSingle",1);

} //end if
else
{
if (OpReqd == 1)
{
//printf("OpReqd = 1 \n");
OpType=2;
SetTagBit("ASRSOpCompDual",1);
ASRSOp();
}//end if
else
{
if (OpReqd == 2)
{
//printf("OpReqd = 2 \n");
OpType=1;
SetTagBit("ASRSOpCompDual",1);
ASRSOp();
}//end if
}//end else
}// end else

}//end if Dual

printf("End ASRSOpComp \n");

}
```

ASRSDwell

```
#include "apdefap.h"

void ASRSDwell()
{
long xActual, yActual, xReqd, yReqd;
extern long OpReqd;
extern int XCo, YCo;

printf("Start ASRSDwell \n");


//SetTagBit("Homepos2",1);
if (GetTagBit("DwellPickPoint") ==1)
{
SetTagWord("xAUTOReqPos",4);
SetTagWord("yAUTOReqPos",3);
printf("Pick Point Selected \n");

}// end if

if (GetTagBit("DwellDepositPoint") == 1)
{
SetTagWord("xAUTOReqPos",5);
SetTagWord("yAUTOReqPos",3);
printf("Deposit Point Selected \n");
}//end if

if (GetTagBit("CurrentLocation")==1)
{
xActual=GetTagWord("xAUTOActPos");
yActual=GetTagWord("yAUTOActPos");
SetTagWord("xAUTOReqPos",xActual);
SetTagWord("yAUTOReqPos",yActual);

printf("Current Location Selected \n");
}//end if

if (GetTagBit("Origin") == 1)
{

SetTagWord("xAUTOReqPos",0);
SetTagWord("yAUTOReqPos",0);
printf ("Origin Selected \n");
}// end if

if (GetTagBit("UserDefined") == 1)
{
xReqd=GetTagWord("xReqdPos");
yReqd=GetTagWord("yReqdPos");
SetTagWord("xAUTOReqPos",xReqd);
SetTagWord("yAUTOReqPos",yReqd);

printf ("User Defined Selected \n");
}// end if

SetTagBit("DwellComplete",1);

if(GetTagBit("ContinuousOperation")==1)
```

```
{
SetTagBit("ASRSDwellContinuous",1);
if((GetTagBit("DwellPickPoint")==1)&&(OpReqd == 1) && (GetTagWord("xAUTOActPos")==4)
&& (GetTagWord("yAUTOActPos")==3))
{
SetTagBit("ASRSDwellAtXYPosition",1);
}//end if
else
if((GetTagBit("CurrentLocation")==1)&&(OpReqd == 2) &&
(GetTagWord("xAUTOActPos")==XCo) && (GetTagWord("yAUTOActPos")==YCo))
{
SetTagBit("ASRSDwellAtXYPosition",1);
}//end if
else
if((GetTagBit("UserDefined")==1)&&(OpReqd == 2) && (GetTagWord("xAUTOActPos")==XCo)
&& (GetTagWord("yAUTOActPos")==YCo))
{
SetTagBit("ASRSDwellAtXYPosition",1);
}//end if



else
{
SetTagBit("GoToXY",1);
}

ASRSOp();
}

else
{
SetTagBit("ASRSDwellNotContinuous",1);
if((GetTagBit("DwellPickPoint")==1)&&(OpReqd == 1) && (GetTagWord("xAUTOActPos")==4)
&& (GetTagWord("yAUTOActPos")==3))
{
SetTagBit("AtXYPosition",1);
}//end if
else
if((GetTagBit("CurrentLocation")==1)&&(OpReqd == 2) &&
(GetTagWord("xAUTOActPos")==XCo) && (GetTagWord("yAUTOActPos")==YCo))
{
SetTagBit("AtXYPosition",1);
}//end if
else
if((GetTagBit("UserDefined")==1)&&(OpReqd == 2) && (GetTagWord("xAUTOActPos")==XCo)
&& (GetTagWord("yAUTOActPos")==YCo))
{
SetTagBit("AtXYPosition",1);
}//end if



else
{
SetTagBit("GoToXY",1);
}

}
//SetTagBit("GoToXY",0);
```

```
printf("End ASRSDwell \n");

}
```

GoToXYCompleteAction

```
#include "apdefap.h"

int gscAction( void )
{


if (GetTagBit("AtXYPosition")==1)
{
printf("Start GoToXYCompleteAction \n");

//printf("PickReqd = %d \n", (GetTagBit("PickReqd")));
//printf("RunPick = %d \n", (GetTagBit("RunPick")));
//printf("PlaceReqd = %d \n", (GetTagBit("PlaceReqd")));
//printf("RunPlace = %d \n", (GetTagBit("RunPlace")));
//printf("DwellComplete = %d \n", (GetTagBit("DwellComplete")));

//printf("Got This Far in GoToXYCompleteAction \n");

if ((GetTagBit("PickReqd")==1)&&(GetTagBit("ASRSOpStart")==1))
{
if (GetTagBit("ForksOn")==1)
{
printf("ForksOn Selected \n");
SetTagBit("RunPick",1);
//printf("PickReqd is set to 1in GoToXYCompleteAction \n");
SetTagBit("PickReqd",0);
SetTagBit("ASRSOpStart",0);
SetTagBit("PlaceCompletePickReqd",0);

}// end if ForksOn
else
{
SetTagBit("RunPick",1);
printf("PictReqd is set to 1in GoToXYCompleteAction \n");
SetTagBit("PickReqd",0);
SetTagBit("ASRSOpStart",0);
SetTagBit("PlaceCompletePickReqd",0);

}// end else
}//end if PickReqd
if((GetTagBit("PickReqd")==1)&&(GetTagBit("PlaceCompletePickReqd")==1))
{
if (GetTagBit("ForksOn")==1)
{
printf("ForksOn Selected \n");
SetTagBit("RunPick",1);
//printf("PickReqd is set to 1in GoToXYCompleteAction \n");
SetTagBit("PickReqd",0);
SetTagBit("ASRSOpStart",0);
SetTagBit("PlaceCompletePickReqd",0);

}// end if ForksOn
else
{
SetTagBit("RunPick",1);
printf("PictReqd is set to 1in GoToXYCompleteAction \n");
SetTagBit("PickReqd",0);
SetTagBit("ASRSOpStart",0);
SetTagBit("PlaceCompletePickReqd",0);

}// end else
}//end if PickReqd

if((GetTagBit("PickReqd")==1)&&(GetTagBit("GoToXYCompleteActionPick")==1))
{
if (GetTagBit("ForksOn")==1)
{
printf("ForksOn Selected \n");
SetTagBit("RunPick",1);
//printf("PickReqd is set to 1in GoToXYCompleteAction \n");
SetTagBit("PickReqd",0);
SetTagBit("ASRSOpStart",0);
SetTagBit("GoToXYCompleteActionPick",0);

}// end if ForksOn
else
{
SetTagBit("RunPick",1);
printf("PictReqd is set to 1in GoToXYCompleteAction \n");
SetTagBit("PickReqd",0);
SetTagBit("ASRSOpStart",0);
SetTagBit("GoToXYCompleteActionPick",0);

}// end else
}//end if PickReqd

if ((GetTagBit("PlaceReqd")==1)&&(GetTagBit("SetPlaceCoOrdsPlaceReqd")==1))
{
if (GetTagBit("ForksOn")==1)
{
printf("ForksOn Selected \n");
SetTagBit("RunPlace",1);
SetTagBit("PlaceReqd",0);
SetTagBit("SetPlaceCoOrdsPlaceReqd",0);
ASRSOpComp();

}// end if ForksOn
else
{
SetTagBit("RunPlace",1);
SetTagBit("PlaceReqd",0);
SetTagBit("SetPlaceCoOrdsPlaceReqd",0);
ASRSOpComp();

}// end else
}//end if PlaceReqd

if ((GetTagBit("DwellComplete")==1)&&(GetTagBit("ASRSDwellNotContinuous")==1))
{
SetTagBit("DwellComplete",0);
SetTagBit("ASRSDwellNotContinuous",0);
}//end if DwellComplete

if
((GetTagBit("DwellComplete")==1)&&(GetTagBit("ASRSOpContinuous")==1)&&(GetTagBit("ASR
SOpNoRequirements")==0))
{
SetTagBit("DwellComplete",0);
SetTagBit("ASRSOpContinuous",0);
SetTagBit("GoToXYCompleteActionPick",1);
ASRSOpAction();
SetTagBit("GoToXY",1);
}

if
((GetTagBit("DwellComplete")==1)&&(GetTagBit("ASRSOpContinuous")==1)&&(GetTagBit("ASR
SOpNoRequirements")==1))
{
SetTagBit("DwellComplete",0);
SetTagBit("ASRSOpContinuous",0);
SetTagBit("ASRSOpNoRequirements",0);
//SetTagBit("GoToXYCompleteActionPick",1);
//ASRSOpAction();
//SetTagBit("GoToXY",1);
}


//printf("PickReqd = %d \n", (GetTagBit("PickReqd")));
//printf("RunPick = %d \n", (GetTagBit("RunPick")));
//printf("PlaceReqd = %d \n", (GetTagBit("PlaceReqd")));
//printf("RunPlace = %d \n", (GetTagBit("RunPlace")));
//printf("DwellComplete = %d \n", (GetTagBit("DwellComplete")));
SetTagBit("AtXYPosition",0);
printf("End GoToXYCompleteAction \n");

}//end if AtXYPosition

// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END

return 0;
}
```

PickComplete

```
#include "apdefap.h"

int gscAction( void )
{
//printf("PickComplete = %d \n", (GetTagBit("PickComplete")));

if (GetTagBit("PickComplete")==1)
{
printf("Start PickComplete \n");
SetTagBit ("PickComplete",0);
SetPlaceCoOrds();
printf("End PickComplete \n");

}
// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END

return 0;
}
```

PlaceComplete

```
#include "apdefap.h"

int gscAction( void )
{
extern long OpReqd;
extern int XCo, YCo;

if (GetTagBit("PlaceComplete")==1)
{
if(GetTagBit("ASRSOpOpTypeNotAvailable")==0)
{
printf("Start PlaceComplete \n");
SetTagBit("PlaceComplete",0);
SetTagBit("ASRSOpDual",0);
SetTagBit("RunPlace",0);
SetTagBit("PlaceCompletePickReqd",1);
SetTagBit("PickReqd",1);
if((GetTagBit("DwellPickPoint")==1)&&(OpReqd == 1) && (GetTagWord("xAUTOActPos")==4)
&& (GetTagWord("yAUTOActPos")==3))
{
SetTagBit("AtXYPosition",1);
}//end if
else
if((GetTagBit("CurrentLocation")==1)&&(OpReqd == 2) &&
(GetTagWord("xAUTOActPos")==XCo) && (GetTagWord("yAUTOActPos")==YCo))
{
SetTagBit("AtXYPosition",1);
}//end if
else
if((GetTagBit("UserDefined")==1)&&(OpReqd == 2) && (GetTagWord("xAUTOActPos")==XCo)
&& (GetTagWord("yAUTOActPos")==YCo))
{
SetTagBit("AtXYPosition",1);
}//end if



else
{
SetTagBit("GoToXY",1);
}

printf("End PlaceComplete \n");
}

if(GetTagBit("ASRSOpOpTypeNotAvailable")==1)
{
SetTagBit("ASRSOpOpTypeNotAvailable",0);
ASRSDwell();
}

}
// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END

return 0;
}
```

PlaceComplete2

```
#include "apdefap.h"

int gscAction( void )
{
//printf("PickComplete = %d \n", (GetTagBit("PickComplete")));

if (GetTagBit("PlaceComplete2")==1)
{
printf("Start PlaceComplete2 \n");
SetTagBit ("PlaceComplete2",0);
SetTagBit("ASRSOpCompSingle",0);
SetTagBit("RunPlace",0);
SetTagBit("PlaceComplete2DwellReqd",1);
ASRSDwell();
printf("End PlaceComplete2 \n");

}

// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END

return 0;
}
```

Start Button

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{

extern long OpType;



SetTagBit("PickReqd",0);
SetTagBit("RunPick",0);
SetTagBit("PickComplete",0);
SetTagBit("TimeDelay",0);

SetTagBit("PlaceReqd",0);
SetTagBit("RunPlace",0);
SetTagBit("PlaceComplete",0);

SetTagBit("GoToXY",0);
SetTagBit("LatchGoToXYSimRect",0);
SetTagBit("AtXYPosition",0);

SetTagBit("DwellComplete",0);
SetTagBit("ASRSOpStart",0);
SetTagBit("ASRSDwellNotContinuous",0);
SetTagBit("ASRSOpContinuous",0);
SetTagBit("SetPlaceCoOrdsPlaceReqd",0);
SetTagBit("PlaceCompletePickReqd",0);
SetTagBit("GoToXYCompleteActionPick",0);
SetTagBit("PlaceComplete2",0);
SetTagBit("GoToXYManual",0);

printf("Starting Operation \n");

OpType = 0;
SetTagBit("Start",1);

ASRSOp();


// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END

}
```

Home and Reset Button

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{
SetTagBit("Homepos2",1);

SetTagBit("PickReqd",0);
SetTagBit("RunPick",0);
SetTagBit("PickComplete",0);

SetTagBit("PlaceReqd",0);
SetTagBit("RunPlace",0);
SetTagBit("PlaceComplete",0);

SetTagBit("GoToXY",0);
SetTagBit("LatchGoToXYSimRect",0);
SetTagBit("AtXYPosition",0);

SetTagBit("DwellComplete",0);
SetTagBit("ASRSOpStart",0);
SetTagBit("ASRSDwellNotContinuous",0);
SetTagBit("ASRSOpContinuous",0);
SetTagBit("SetPlaceCoOrdsPlaceReqd",0);
SetTagBit("PlaceCompletePickReqd",0);
SetTagBit("GoToXYCompleteActionPick",0);
SetTagBit("PlaceComplete2",0);

// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END

}
```

GoTo Dwell

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{
long xActual, yActual, xReqd, yReqd;

SetTagBit("PickReqd",0);
SetTagBit("AtXYPosition",0);
SetTagBit("RunPick",0);
SetTagBit("PlaceReqd",0);
SetTagBit("RunPlace",0);
SetTagBit("DwellComplete",0);
SetTagBit("PickComplete",0);
SetTagBit("PlaceComplete",0);
SetTagBit("PickReqd",0);
SetTagBit("ASRSOpStart",0);
SetTagBit("ASRSDwellNotContinuous",0);
SetTagBit("ASRSOpContinuous",0);
SetTagBit("SetPlaceCoOrdsPlaceReqd",0);
SetTagBit("PlaceCompletePickReqd",0);
SetTagBit("GoToXYCompleteActionPick",0);
SetTagBit("PlaceComplete2",0);
SetTagBit("GoToXYManual",1);
//SetTagBit("Homepos2",1);
if (GetTagBit("DwellPickPoint") ==1)
{
SetTagWord("xAUTOReqPos",4);
SetTagWord("yAUTOReqPos",3);
printf("Pick Point Selected \n");

}// end if

if (GetTagBit("DwellDepositPoint") == 1)
{
SetTagWord("xAUTOReqPos",5);
SetTagWord("yAUTOReqPos",3);
printf("Deposit Point Selected \n");
}//end if

if (GetTagBit("CurrentLocation")==1)
{
xActual=GetTagWord("xAUTOActPos");
yActual=GetTagWord("yAUTOActPos");
SetTagWord("xAUTOReqPos",xActual);
SetTagWord("yAUTOReqPos",yActual);

printf("Current Location Selected \n");
}//end if

if (GetTagBit("Origin") == 1)
{

SetTagWord("xAUTOReqPos",0);
SetTagWord("yAUTOReqPos",0);
printf ("Origin Selected \n");
}// end if

if (GetTagBit("UserDefined") == 1)
{
xReqd=GetTagWord("xReqdPos");
yReqd=GetTagWord("yReqdPos");
SetTagWord("xAUTOReqPos",xReqd);
SetTagWord("yAUTOReqPos",yReqd);

printf ("User Defined Selected \n");
}// end if

SetTagBit("GoToXY",1);
//SetTagBit("GoToY",1);


// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END

}
```

GoTo X

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{
long xReqd, yReqd;

xReqd=GetTagWord("xReqdPos");
yReqd=GetTagWord("yReqdPos");
SetTagWord("xAUTOReqPos",xReqd);
SetTagWord("yAUTOReqPos",yReqd);

SetTagBit("GoToX",1);


// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END

}
```

GoTo Y

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{

long xReqd, yReqd;

xReqd=GetTagWord("xReqdPos");
yReqd=GetTagWord("yReqdPos");
SetTagWord("xAUTOReqPos",xReqd);
SetTagWord("yAUTOReqPos",yReqd);


SetTagBit("GoToY",1);


// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END

}
```

GoToXY

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName, char* lpszPropertyName,
UINT nFlags, int x, int y)
{
long xReqd, yReqd;

SetTagBit("PickReqd",0);
SetTagBit("AtXYPosition",0);
SetTagBit("RunPick",0);
SetTagBit("PlaceReqd",0);
SetTagBit("RunPlace",0);
SetTagBit("DwellComplete",0);
SetTagBit("PickComplete",0);
SetTagBit("PlaceComplete",0);
SetTagBit("PickReqd",0);
SetTagBit("ASRSOpStart",0);
SetTagBit("ASRSDwellNotContinuous",0);
SetTagBit("ASRSOpContinuous",0);
SetTagBit("SetPlaceCoOrdsPlaceReqd",0);
SetTagBit("PlaceCompletePickReqd",0);
SetTagBit("GoToXYCompleteActionPick",0);
SetTagBit("PlaceComplete2",0);

xReqd=GetTagWord("xReqdPos");
yReqd=GetTagWord("yReqdPos");
SetTagWord("xAUTOReqPos",xReqd);
SetTagWord("yAUTOReqPos",yReqd);

SetTagBit("GoToXYManual",1);
SetTagBit("GoToXY",1);
//SetTagBit("GoToY",1);


// WINCC:TAGNAME_SECTION_START
// syntax: #define TagNameInAction "DMTagName"
// next TagID : 1
// WINCC:TAGNAME_SECTION_END

// WINCC:PICNAME_SECTION_START
// syntax: #define PicNameInAction "PictureName"
// next PicID : 1
// WINCC:PICNAME_SECTION_END

}
```

Appendix D (Database Tables)

Field Name	Data Type
BayNumber (Primary Key)	Number (Long Integer)
XCoOrdinate	Number (Long Integer)
YCoOrdinate	Number (Long Integer)
Dedicated	Number (Long Integer)
Class	Number (Long Integer)
ComponentID	Number (Long Integer)
DateIn	Number (Double)
Checked	Number (Long Integer)
Date/Time	Date/Time
KittingZone	Number (Long Integer)
ObsoleteZone	Number (Long Integer)

Table D-1: ASRSContents Fields

Field Name	Data Type
ComponentID (Primary Key)	Number (Long Integer)
Class	Number (Long Integer)

Table D-2: ClassType Fields

Field Name	Data Type
OpNumber (Primary Key)	Number (Long Integer)
InterOpTime	Number (Double)

Table D-3: InterTimes Fields

Field Name	Data Type
OperationNumber (Primary Key)	Number (Long Integer)
PartNumber	Number (Long Integer)
OperationComp	Number (Long Integer)
DateTime	Number (Long Integer)

Table D-4: OperationsCompleted Fields

Field Name	**Data Type**
OpNumber (Primary Key)	Number (Long Integer)
PartNumber	Number (Long Integer)
OperationReqd	Number (Long Integer)
DateTimeReqd	Number (Double)
Checked	Number (Long Integer)

Table D-5: Requirements Fields

Field Name	**Data Type**
OpNumber (Primary Key)	Number (Long Integer)
PartNumber	Number (Long Integer)
OperationReqd	Number (Long Integer)
DateTimeReqd	Number (Double)
Checked	Number (Long Integer)

Table D-6: tmpRequirements Fields

Field Name	**Data Type**
BayNumber (Primary Key)	Number (Long Integer)
XCoOrdinate	Number (Long Integer)
YCoOrdinate	Number (Long Integer)
Dedicated	Number (Long Integer)
Class	Number (Long Integer)
ComponentID	Number (Long Integer)
DateIn	Number (Double)
Checked	Number (Long Integer)
Date/Time	Date/Time
KittingZone	Number (Long Integer)
ObsoleteZone	Number (Long Integer)

Table D-7: tmpASRSContents Fields

Appendix E (List of SCADA Tags and PLC Variables)

List of Variables

Address	Name	Where Used			
C1	Count X	FC15	FC17	FC18	
C2	Count Y	FC16	FC17	FC18	
I0.0	X Reflective Sensor	FC15	FC17		
I0.1	Forks Retracted	FC23			
I0.3	Y Reflective Sensor	FC16			
I0.6	Forks Extended	FC23			
I0.7	Pallet Sensor				
I1.0	X Forward Limit				
I1.3	Y Home	FC14	FC16	FC17	FC23
I1.4	Y Upper Limit	FC23			
I2.0	Manual	OB1	FC1		
I2.1	X Forward Manual	FC1			
I2.2	X Reverse Manual	FC1			
I2.3	Y Drive On Manual	FC1			
I2.4	Y High Speed Manual	FC1			
I3.2	X Home	FC14	FC15	FC17	
M0.2	YManualOnTag_1	WinCC Tag			
M0.3	YManualOnTag_2	WinCC Tag			
M0.4	AtXYPosition	WinCC Tag			
M0.5	AtDwellPoint	WinCC Tag			
M1.5	Flag T40 Complete	FC24			
M5.4	3.7 Sec. Complete	FC23	FC25		
M5.5	2.9 Sec. Complete	FC23			
M6.1	Neg. Edge T10	FC23			
M7.3		FC23			
M7.6	Neg. Edge T20	FC23			
M6.2	Flag Forks Retracted	FC24			
M6.5	Flag T30 Complete	FC24			
M7.2	Neg. Edge M6.5	FC24			
M100.0	ASRSLeft	WinCC Tag			
M100.2	ASRSDown	WinCC Tag			
M100.3	ASRSUp	WinCC Tag			
M101.0	ForksExtend	WinCC Tag	FC21	OB1	
M101.1	ForksRetract	WinCC Tag	FC22	OB1	
M101.2	Forks Extend Latch	OB1	FC21		
M101.3	Forks Retract Latch	OB1	FC22		
M101.5	Forks Extend End	FC21			
M101.6	Forks Retract End	FC22			
M102.0	go_1	WinCC Tag			
m102.1	home_1	WinCC Tag			
M103.0	go_2	WinCC Tag			
M103.1	RESET	WinCC Tag			
M103.2	Homepos2	WinCC Tag	FC14		
M104.1	TimeDelay	WinCC Tag	OB1		
M104.2	Time Delay Latch	OB1	FC2		

M104.3	30SecsComplete	WinCC Tag			
M104.4	30 Secs. Over	FC2			
M112.4	Homepos2 Latch	FC14	OB1		
M112.5	At Home	FC14			
M201.0	FIFO	WinCC Tag			
M201.1	Random_1	WinCC Tag			
M202.0	DwellPickPoint	WinCC Tag			
M202.1	DwellDepositPoint	WinCC Tag			
M202.2	CurrentLocation	WinCC Tag			
M202.3	Origin	WinCC Tag			
M202.4	UserDefined	WinCC Tag			
M203.0	Rectilinear	WinCC Tag			
M203.1	Simultaneous	WinCC Tag			
M203.2	GoToX	WinCC Tag	OB1		
M203.3	GoToY	WinCC Tag	OB1		
M203.4	Go to X Latch	OB1	FC15		
M203.5	Go to Y Latch	OB1	FC16		
M203.6	GoToXY	WinCC Tag	OB1	FC17	FC18
M203.7	LatchGoToXYSimRect	WinCC Tag	OB1		
M204.0	Single	WinCC Tag			
M204.1	Dual	WinCC Tag			
M204.2	CMP==1 X	FC17			
M204.3	CMP==1 Y	FC17			
M204.4	CMP==1 X	FC18			
M205.0	ForksOn	WinCC Tag			
M205.1	ForksOff	WinCC Tag			
M205.2	BufferZoneRetrieval	WinCC Tag			
M205.3	BufferZoneStorage	WinCC Tag			
M205.4	ObsoleteZoneON	WinCC Tag			
M205.5	ObsoleteZoneOFF	WinCC Tag			
M210.0	Fc Complete	FC14 - FC20			
M210.1	20 ms Complete	FC16	FC17	FC18	
M300.0	Random	WinCC Tag			
M300.1	nearest	WinCC Tag			
M300.2	Dedicated	WinCC Tag			
M300.3	Classbased	WinCC Tag			
M300.4	Free	WinCC Tag			
M301.0	nearest_1	WinCC Tag			
M302.0	TimeControl	WinCC Tag			
M310.2	xAUTOPos_3	WinCC Tag			
M310.3	xAUTOPos_4	WinCC Tag			
M310.4	xAUTOPos_5	WinCC Tag			
M310.5	xAUTOPos_6	WinCC Tag			
M310.6	xAUTOPos_7	WinCC Tag			
M310.7	xAUTOPos_8	WinCC Tag			
M311.2	yAUTOPos_3	WinCC Tag			
M311.3	yAUTOPos_4	WinCC Tag			

M311.4	yAUTOPos_5	WinCC Tag			
M311.5	yAUTOPos_6	WinCC Tag			
M311.6	yAUTOPos_7	WinCC Tag			
M311.7	yAUTOPos_8	WinCC Tag			
M500.1	ASRSright	WinCC Tag			
M600.0	Homepos	WinCC Tag			
M780.2	Neg. Edge Y	FC16	FC17		
M790.2	Neg. Edge X	FC15	FC17		
M888.0	TagWait	WinCC Tag			
M888.6	LiamTrigger1	WinCC Tag			
M888.7	LiamTrigger	WinCC Tag			
M943.0		FC16	FC17		
M943.4		FC15	FC17		
M946.2	CMP < 1 X	FC15	FC17		
M946.3	CMP > 1 X	FC15	FC17		
M946.4	CMP < 1 Y	FC16	FC17		
M946.5	CMP > 1 Y	FC16	FC17		
M950.0	Store	WinCC Tag	FC23	OB1	
M950.1	Retrieve	WinCC Tag	OB1	FC24	FC25
M950.3	Retrieve Latch	OB1	FC24		
M951.0	FC25 Flag	FC23	FC25		
M951.1	Store Latch	OB1	FC23		
M952.0	Stop StoreRetreive	FC23	FC24		
M960.0	FC 25 Set	FC25			
M960.2	FC25 Flag	FC25			
M960.3	Forks Retracted 1	FC25			
M960.4	Forks Retracted 2	FC25			
M978.0	classbased_update	WinCC Tag			
M978.1	dedicated_update	WinCC Tag			
M978.2	ASRSlayout_update	WinCC Tag			
M1100.0	PickReqd	WinCC Tag			
M1100.1	RunPick	WinCC Tag	FC19	OB1	
M1100.2	PickComplete	WinCC Tag			
M1100.3	Run Pick Latch	OB1	FC19		
M1100.4	T1 Complete	FC19			
M1101.0	PlaceReqd	WinCC Tag			
M1101.1	RunPlace	WinCC Tag	OB1	FC20	
M1101.2	PlaceComplete	WinCC Tag			
M1101.3	Run Place Latch	OB1	FC20		
M1102.0	DwellReqd	WinCC Tag			
M1102.1	DwellComplete	WinCC Tag			
M1102.2	JobComplete	WinCC Tag			
M1102.3	ContinuousOperation	WinCC Tag			
M1103.0	Start	WinCC Tag			
M1103.1	ASRSOpStart	WinCC Tag			
M1103.2	ASRSOpCompDual	WinCC Tag			
M1103.3	ASRSOpCompSingle	WinCC Tag			
M1103.4	ASRSOpDual	WinCC Tag			

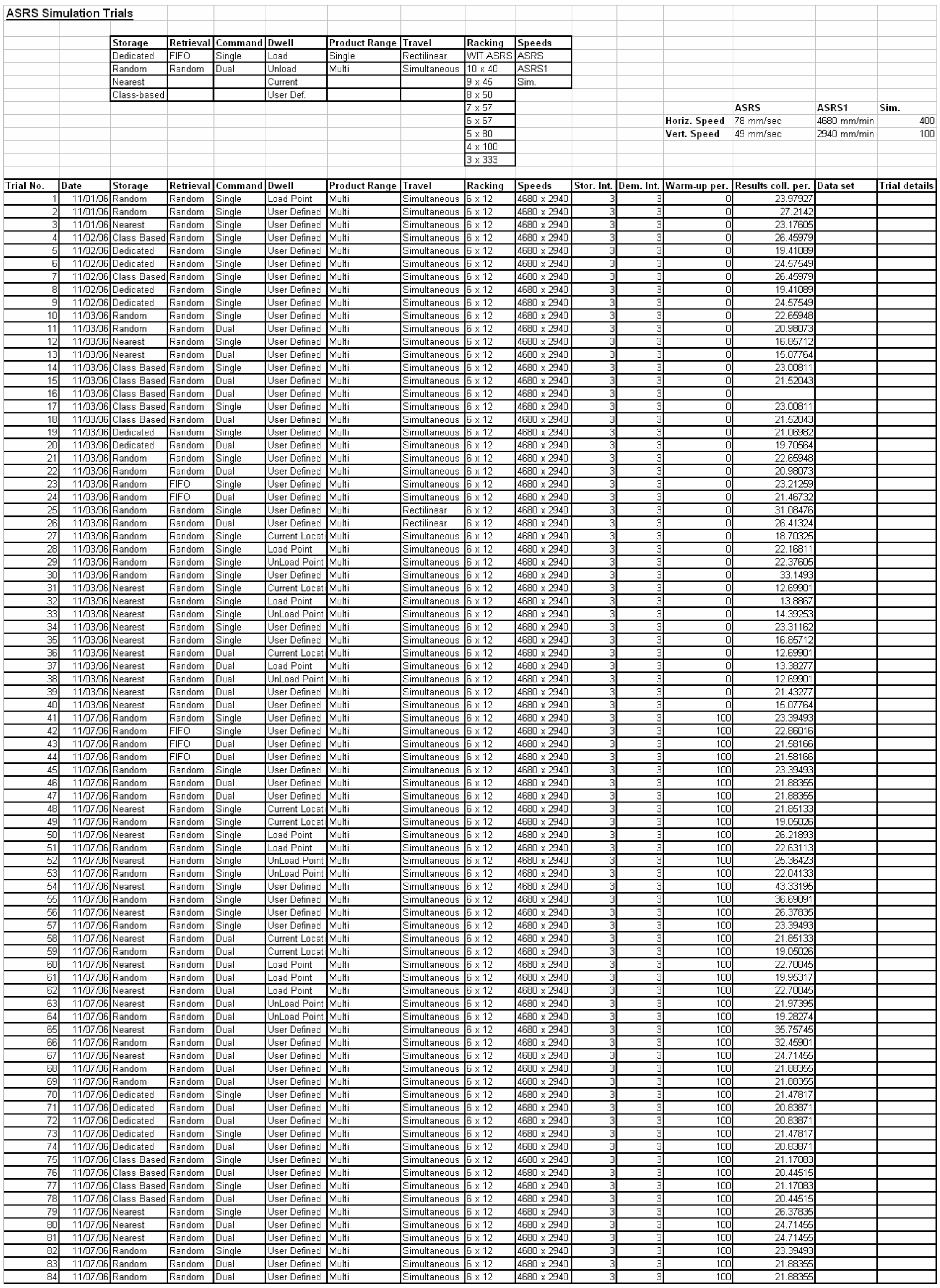
M1103.5	PlaceComplete1	WinCC Tag			
M1103.6	PlaceComplete2	WinCC Tag			
M1103.7	ASRSDwellContinuous	WinCC Tag			
M1104.0	ASRSDwellNotContinuous	WinCC Tag			
M1104.1	ASRSOpContinuous	WinCC Tag			
M1104.2	SetPlaceCoOrdsPlaceReqd	WinCC Tag			
M1104.3	PlaceCompletePickReqd	WinCC Tag			
M1104.4	PlaceComplete2DwellReqd	WinCC Tag			
M1104.5	GoToXYCompleteActionPick	WinCC Tag			
M1104.6	GoToXYManual	WinCC Tag			
M1104.7	ASRSOpOpTypeNotAvailable	WinCC Tag			
M1105.0	ASRSOpNoRequirements	WinCC Tag			
M1105.1	ASRSDwellAtXYPosition	WinCC Tag			
M1200.0	Callauto	WinCC Tag			
M1200.1	CallSemiauto	WinCC Tag			
M1200.2	Store2Tag	WinCC Tag			
M1200.3	go3	WinCC Tag			
M1200.4	RetrieveTag	WinCC Tag			
M1200.5	Storenew	WinCC Tag			
M1200.6	Retrievenew	WinCC Tag			
MW1000	classbased1-2	WinCC Tag			
MW1002	classbased1-1	WinCC Tag			
MW1004	classbased1-3	WinCC Tag			
MW1006	classbased1-4	WinCC Tag			
MW1008	classbased1-5	WinCC Tag			
MW1010	classbased1-6	WinCC Tag			
MW1012	classbased2-1	WinCC Tag			
MW1014	classbased2-2	WinCC Tag			
MW1016	classbased2-3	WinCC Tag			
MW1018	classbased2-4	WinCC Tag			
MW1020	classbased2-5	WinCC Tag			
MW1022	dedicated1-1	WinCC Tag			
MW1024	dedicated1-2	WinCC Tag			
MW1026	dedicated1-5	WinCC Tag			
MW1028	dedicated1-6	WinCC Tag			
MW1030	classbased2-6	WinCC Tag			
MW1032	dedicated2-1	WinCC Tag			
MW1034	dedicated2-2	WinCC Tag			
MW1036	dedicated2-3	WinCC Tag			
MW1038	dedicated2-4	WinCC Tag			
MW1040	dedicated2-5	WinCC Tag			
MW1042	dedicated2-6	WinCC Tag			
MW1044	ASRSlayout1-1	WinCC Tag			
MW1046	ASRSlayout1-2	WinCC Tag			
MW1048	ASRSlayout1-3	WinCC Tag			
MW1050	ASRSlayout1-4	WinCC Tag			
MW1052	ASRSlayout1-5	WinCC Tag			
MW1054	ASRSlayout1-6	WinCC Tag			

MW1056	ASRSlayout2-1	WinCC Tag			
MW1058	ASRSlayout2-2	WinCC Tag			
MW1060	ASRSlayout2-4	WinCC Tag			
MW1062	ASRSlayout2-3	WinCC Tag			
MW1064	ASRSlayout2-5	WinCC Tag			
MW1068	ASRSlayout2-6	WinCC Tag			
MW1070	dedicated1-3	WinCC Tag			
MW1072	dedicated1-4	WinCC Tag			
MW1080	SemiautoXReqPos	WinCC Tag			
MW1082	SemiautoYReqPos	WinCC Tag			
MW1084	TestTag	WinCC Tag			
MW1086	XCoOrdinateDB	WinCC Tag			
MW1088	YCoOrdinateDB	WinCC Tag			
MW1090	autocomponentID	WinCC Tag			
MW1210	Liam1	WinCC Tag			
MW1212	Liam2	WinCC Tag			
MW1214	GotThisFar	WinCC Tag			
MW1216	OpReqdOP	WinCC Tag			
MW1218	OpNumberOP	WinCC Tag			
MW312	OPreq_1	WinCC Tag			
MW316	Opreq_2	WinCC Tag			
MW320	OPreq_3	WinCC Tag			
MW324	OPreq_4	WinCC Tag			
MW328	OPreq_5	WinCC Tag			
MW332	OPreq_6	WinCC Tag			
MW336	OPreq_7	WinCC Tag			
MW340	OPreq_8	WinCC Tag			
MW344	xAUTOPos_1	WinCC Tag			
MW348	yAUTOPos_1	WinCC Tag			
MW352	xAUTOPos_2	WinCC Tag			
MW356	yAUTOPos_2	WinCC Tag			
MW700	xReqPos	WinCC Tag			
MW702	YReqPos	WinCC Tag			
MW710	XActPosn	WinCC Tag			
MW720	YActPosn	WinCC Tag			
MW725	yAUTOActPos	WinCC Tag	FC16	FC17	
MW730	yReqdPos	WinCC Tag			
MW735	yAUTOReqPos	WinCC Tag	FC16	FC17	
MW745	xAUTOActPos	WinCC Tag	FC15	FC17	
MW750	xReqdPos	WinCC Tag			
MW755	xAUTOReqPos	WinCC Tag	FC15	FC17	
Q0.0	Y Drive On	FC14	FC16	FC17	
Q0.1	Y High Speed	FC14	FC16	FC17	
Q0.2	Y Drive Down	FC14	FC16	FC17	FC23
Q0.5	X Forward	FC15	FC17		
Q0.6	X Reverse	FC14	FC15	FC17	
Q0.7	Forks On	FC23			
Q3.0	Forks Extend	FC23			

Q3.1	Forks Retract	FC23			
T1	20 ms Pulse on Y Home	FC16	FC17		
T10	3.7 Sec. Timer	FC23			
T20	2.9 Sec. Timer	FC23			
T30	2.1 Sec. Timer	FC24			
T40	0.08 Sec. Timer	FC24			
T41	5 Sec. Timer	FC2			
T50	2.5 Sec. Timer	Fc25			

Appendix F (Trial Results)

ASRS Trials											
							Physical Model		**Mathematical Model**		
Trial No.	**Req. Type**	**Dwell Point**	**Movement**	**Single / Dual**	**Storage**	**Retrieval**	**Average**	**Comments**	**Sim Results**	**Average**	**Diff (%)**
1	40 Store	Pick Point	Simult.	Single	Free-Random	Random	**1393**		24.0	1438.8	-3.3
2	40 Store	(6,3)	Simult.	Single	Free-Random	Random	**1613**		27.2	1632.9	-1.3
3	40 Store	(6,3)	Simult.	Single	Free-Nearest	Random	**1439**		23.2	1390.6	3.4
4	40 Store	(6,3)	Simult.	Single	Class-Random	Random	**1493**		26.5	1587.6	-6.3
5	40 Store	(6,3)	Simult.	Single	Dedicated-Random	Random	**1390**		19.4	1411.7	-1.6
6	40 Store	(6,3)	Simult.	Single	Dedicated-Random	Random	**1477**		24.6	1474.5	0.2
7	40 Store	(6,3)	Simult.	Single	Class-Nearest	Random	**1359**	3rd trial slow due to internet access	26.5	1587.6	-16.8
8	40 Store	(6,3)	Simult.	Single	Dedicated-Nearest	Random	**1321**		19.4	1411.7	-6.8
9	40 Store	(6,3)	Simult.	Single	Dedicated-Nearest	Random	**1395**		24.6	1474.5	-5.7
10	23S / 17R	(6,3)	Simult.	Single	Free-Random	Random	**1585**	ASRS empty at start (trials 10 - 40)	22.7	1431.1	9.7
11	23S / 17R	(6,3)	Simult.	Dual	Free-Random	Random	**1135**		21.0	1325.1	-16.7
12	23S / 17R	(6,3)	Simult.	Single	Free-Nearest	Random	**0**		16.9	1064.7	
13	23S / 17R	(6,3)	Simult.	Dual	Free-Nearest	Random	**0**		15.1	952.3	
14	23S / 17R	(6,3)	Simult.	Single	Class-Random	Random	**0**		23.0	1453.1	
15	23S / 17R	(6,3)	Simult.	Dual	Class-Random	Random	**0**		21.5	1359.2	
17	23S / 17R	(6,3)	Simult.	Single	Class-Nearest	Random	**0**		23.0	1453.1	
18	23S / 17R	(6,3)	Simult.	Dual	Class-Nearest	Random	**0**		21.5	1359.2	
19	23S / 17R	(6,3)	Simult.	Single	Dedicated-Random	Random	**0**		21.1	1330.7	
20	23S / 17R	(6,3)	Simult.	Dual	Dedicated-Random	Random	**0**		19.7	1244.6	
21	23S / 17R	(6,3)	Simult.	Single	Free-Random	Nearest	**1487**		22.7	1431.1	3.7
22	23S / 17R	(6,3)	Simult.	Dual	Free-Random	Nearest	**1157**		21.0	1325.1	-14.5
23	23S / 17R	(6,3)	Simult.	Single	Free-Random	FIFO	**0**		23.2	1466.1	
24	23S / 17R	(6,3)	Simult.	Dual	Free-Random	FIFO	**0**		21.5	1355.8	
25	23S / 17R	(6,3)	Rect.	Single	Free-Random	Random	**1841**		31.1	1963.2	-6.6
26	23S / 17R	(6,3)	Rect.	Dual	Free-Random	Random	**1447**		26.4	1668.2	-15.3
27	23S / 17R	Current	Simult.	Single	Free-Random	Random	**1262**		18.7	1181.3	6.4
28	23S / 17R	Pick Point	Simult.	Single	Free-Random	Random	**1437**		22.2	1400.1	2.6
29	23S / 17R	Deposit Point	Simult.	Single	Free-Random	Random	**1347**		22.4	1413.2	-4.9
30	23S / 17R	Origin	Simult.	Single	Free-Random	Random	**2499**		33.1	2093.6	16.2
31	23S / 17R	Current	Simult.	Single	Free_Nearest	Nearest	**880**		12.7	802.0	8.9
32	23S / 17R	Pick Point	Simult.	Single	Free_Nearest	Nearest	**955**		13.9	877.1	8.1
33	23S / 17R	Deposit Point	Simult.	Single	Free_Nearest	Nearest	**1008**		14.4	909.0	9.8
34	23S / 17R	Origin	Simult.	Single	Free_Nearest	Nearest	**1910**		23.3	1472.3	22.9
35	23S / 17R	(6,3)	Simult.	Single	Free_Nearest	Nearest	**1144**		16.9	1064.7	6.9
36	23S / 17R	Current	Simult.	Dual	Free_Nearest	Nearest	**731**		12.7	802.0	-9.8
37	23S / 17R	Pick Point	Simult.	Dual	Free_Nearest	Nearest	**738**		13.4	845.2	-14.5
38	23S / 17R	Deposit Point	Simult.	Dual	Free_Nearest	Nearest	**778**		12.7	802.0	-3.1
39	23S / 17R	(0,0)	Simult.	Dual	Free_Nearest	Nearest	**1309**		21.4	1353.6	-3.4
40	23S / 17R	(6,3)	Simult.	Dual	Free_Nearest	Nearest	**883**		15.1	952.3	-7.9
41	20S / 20R	(6,3)	Simult.	Single	Free-Random	Random	**1510**	Fully loaded (Dedicated) at Start	23.4	1559.7	-3.3
42	20S / 20R	(6,3)	Simult.	Single	Free-Random	FIFO	**1440**		22.9	1524.0	-5.8
43	20S / 20R	(6,3)	Simult.	Dual	Free-Random	FIFO	**1306**	Retrieval - Storage	21.6	1438.8	-10.1
44	20S / 20R	(6,3)	Simult.	Dual	Free-Random	FIFO	**1271**	Storage - Retrieval	21.6	1438.8	-13.2
45	20S / 20R	(6,3)	Simult.	Single	Free-Random	Nearest	**1003**		23.4	1559.7	-55.5
46	20S / 20R	(6,3)	Simult.	Dual	Free-Random	Nearest	**892**	Retrieval - Storage	21.9	1458.9	-63.5
47	20S / 20R	(6,3)	Simult.	Dual	Free-Random	Nearest	**1125**	Storage - Retrieval	21.9	1458.9	-29.7
48	20S / 20R	Current	Simult.	Single	Free-Nearest	Nearest	**977**		21.9	1456.8	-49.2
49	20S / 20R	Current	Simult.	Single	Free-Random	Random	**1238**		19.1	1270.0	-2.6
50	20S / 20R	Pick Point	Simult.	Single	Free-Nearest	Nearest	**922**		26.2	1747.9	-89.7
51	20S / 20R	Pick Point	Simult.	Single	Free-Random	Random	**1388**		22.6	1508.7	-8.7
52	20S / 20R	Deposit Point	Simult.	Single	Free-Nearest	Nearest	**919**		25.4	1690.9	-84.1
53	20S / 20R	Deposit Point	Simult.	Single	Free-Random	Random	**1476**		22.0	1469.4	0.5
54	20S / 20R	(0,0)	Simult.	Single	Free-Nearest	Nearest	**1654**		43.3	2888.8	-74.7
55	20S / 20R	(0,0)	Simult.	Single	Free-Random	Random	**2420**		36.7	2446.1	-1.1
56	20S / 20R	(6,3)	Simult.	Single	Free-Nearest	Nearest	**1005**		26.4	1758.6	-75.0
57	20S / 20R	(6,3)	Simult.	Single	Free-Random	Random	**1447**		23.4	1559.7	-7.8
58	20S / 20R	Current	Simult.	Dual	Free-Nearest	Nearest	**941**		21.9	1456.8	-54.9
59	20S / 20R	Current	Simult.	Dual	Free-Random	Random	**1108**		19.1	1270.0	-14.7
60	20S / 20R	Pick Point	Simult.	Dual	Free-Nearest	Nearest	**933**		22.7	1513.4	-62.2
61	20S / 20R	Pick Point	Simult.	Dual	Free-Random	Random	**1352**		20.0	1330.2	1.6
62	20S / 20R	Pick Point	Simult.	Dual	Free-Nearest	Nearest	**962**	S - R	22.7	1513.4	-57.3
63	20S / 20R	Deposit Point	Simult.	Dual	Free-Nearest	Nearest	**931**		22.0	1464.9	-57.4
64	20S / 20R	Deposit Point	Simult.	Dual	Free-Random	Random	**1366**		19.3	1285.5	5.9
65	20S / 20R	(0,0)	Simult.	Dual	Free-Nearest	Nearest	**1111**		35.8	2383.8	-114.7
66	20S / 20R	(0,0)	Simult.	Dual	Free-Random	Random	**1677**		32.5	2163.9	-29.1
67	20S / 20R	(6,3)	Simult.	Dual	Free-Nearest	Nearest	**891**		24.7	1647.6	-84.9
68	20S / 20R	(6,3)	Simult.	Dual	Free-Random	Random	**1252**		21.9	1458.9	-16.5
69	20S / 20R	(6,3)	Simult.	Dual	Free-Random	Random	**1290**	S - R	21.9	1458.9	-13.1
70	20S / 20R	(6,3)	Simult.	Single	Dedicated-Random	Nearest	**1226**	ASRS contents equal to dedicated	21.5	1548.0	-26.2
71	20S / 20R	(6,3)	Simult.	Dual	Dedicated-Random	Nearest	**1081**		20.8	1501.9	-38.9
72	20S / 20R	(6,3)	Simult.	Dual	Dedicated-Random	Nearest	**1189**	S - R	20.8	1501.9	-26.3
73	20S / 20R	(6,3)	Simult.	Single	Dedicated-Random	Random	**1441**		21.5	1548.0	-7.4
74	20S / 20R	(6,3)	Simult.	Dual	Dedicated-Random	Random	**1269**		20.8	1501.9	-18.3
75	20S / 20R	(6,3)	Simult.	Single	ClassBased-Random	Nearest	**1186**		21.2	1525.8	-28.7
76	20S / 20R	(6,3)	Simult.	Dual	ClassBased-Random	Nearest	**1032**		20.4	1473.5	-42.8
77	20S / 20R	(6,3)	Simult.	Single	ClassBased-Random	Random	**1358**		21.2	1525.8	-12.4
78	20S / 20R	(6,3)	Simult.	Dual	ClassBased-Random	Random	**1280**		20.4	1473.5	-15.2
79	20S / 20R	(6,3)	Simult.	Single	Free-Nearest	Nearest	**1074**		26.4	1758.6	-63.7
80	20S / 20R	(6,3)	Simult.	Dual	Free-Nearest	Nearest	**959**		24.7	1647.6	-71.9
81	20S / 20R	(6,3)	Simult.	Dual	Free-Nearest	Nearest	**1121**	S - R	24.7	1647.6	-47.0
82	20S / 20R	(6,3)	Simult.	Single	Free-Random	Random	**1480**		23.4	1559.7	-5.4
83	20S / 20R	(6,3)	Simult.	Dual	Free-Random	Random	**1216**		21.9	1458.9	-20.0
84	20S / 20R	(6,3)	Simult.	Dual	Free-Random	Random	**1284**	S - R	21.9	1458.9	-13.6



ASRS Simulation Trials

Storage	Retrieval	Command	Dwell	Product Range	Travel	Racking	Speeds
Dedicated	FIFO	Single	Load	Single	Rectilinear	WIT ASRS	ASRS
Random	Random	Dual	Unload	Multi	Simultaneous	10 x 40	ASRS1
Nearest			Current			9 x 45	Sim.
Class-based			User Def.			8 x 50	
						7 x 57	
						6 x 67	
						5 x 80	
						4 x 100	
						3 x 333	

	ASRS	ASRS1	Sim.
Horiz. Speed	78 mm/sec	4680 mm/min	400
Vert. Speed	49 mm/sec	2940 mm/min	100

Trial No.	Date	Storage	Retrieval	Command	Dwell	Product Range	Travel	Racking	Speeds	Stor. Int.	Dem. Int.	Warm-up per.	Results coll. per.	Data set	Trial details
1	11/01/06	Random	Random	Single	Load Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	23.97927		
2	11/01/06	Random	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	27.2142		
3	11/01/06	Nearest	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	23.17605		
4	11/02/06	Class Based	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	26.45979		
5	11/02/06	Dedicated	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	19.41089		
6	11/02/06	Dedicated	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	24.57549		
7	11/02/06	Class Based	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	26.45979		
8	11/02/06	Dedicated	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	19.41089		
9	11/02/06	Dedicated	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	24.57549		
10	11/03/06	Random	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	22.65948		
11	11/03/06	Random	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	20.98073		
12	11/03/06	Nearest	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	16.85712		
13	11/03/06	Nearest	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	15.07764		
14	11/03/06	Class Based	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	23.00811		
15	11/03/06	Class Based	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	21.52043		
16	11/03/06	Class Based	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0			
17	11/03/06	Class Based	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	23.00811		
18	11/03/06	Class Based	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	21.52043		
19	11/03/06	Dedicated	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	21.06982		
20	11/03/06	Dedicated	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	19.70564		
21	11/03/06	Random	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	22.65948		
22	11/03/06	Random	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	20.98073		
23	11/03/06	Random	FIFO	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	23.21259		
24	11/03/06	Random	FIFO	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	21.46732		
25	11/03/06	Random	Random	Single	User Defined	Multi	Rectilinear	6 x 12	4680 x 2940	3	3	0	31.08476		
26	11/03/06	Random	Random	Dual	User Defined	Multi	Rectilinear	6 x 12	4680 x 2940	3	3	0	26.41324		
27	11/03/06	Random	Random	Single	Current Locati	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	18.70325		
28	11/03/06	Random	Random	Single	Load Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	22.16811		
29	11/03/06	Random	Random	Single	UnLoad Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	22.37605		
30	11/03/06	Random	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	33.1493		
31	11/03/06	Nearest	Random	Single	Current Locati	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	12.69901		
32	11/03/06	Nearest	Random	Single	Load Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	13.8867		
33	11/03/06	Nearest	Random	Single	UnLoad Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	14.39253		
34	11/03/06	Nearest	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	23.31162		
35	11/03/06	Nearest	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	16.85712		
36	11/03/06	Nearest	Random	Dual	Current Locati	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	12.69901		
37	11/03/06	Nearest	Random	Dual	Load Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	13.38277		
38	11/03/06	Nearest	Random	Dual	UnLoad Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	12.69901		
39	11/03/06	Nearest	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	21.43277		
40	11/03/06	Nearest	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	0	15.07764		
41	11/07/06	Random	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	23.39493		
42	11/07/06	Random	FIFO	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	22.86016		
43	11/07/06	Random	FIFO	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.58166		
44	11/07/06	Random	FIFO	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.58166		
45	11/07/06	Random	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	23.39493		
46	11/07/06	Random	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.88355		
47	11/07/06	Random	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.88355		
48	11/07/06	Nearest	Random	Single	Current Locati	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.85133		
49	11/07/06	Random	Random	Single	Current Locati	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	19.05026		
50	11/07/06	Nearest	Random	Single	Load Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	26.21893		
51	11/07/06	Random	Random	Single	Load Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	22.63113		
52	11/07/06	Nearest	Random	Single	UnLoad Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	25.36423		
53	11/07/06	Random	Random	Single	UnLoad Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	22.04133		
54	11/07/06	Nearest	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	43.33195		
55	11/07/06	Random	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	36.69091		
56	11/07/06	Nearest	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	26.37835		
57	11/07/06	Random	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	23.39493		
58	11/07/06	Nearest	Random	Dual	Current Locati	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.85133		
59	11/07/06	Random	Random	Dual	Current Locati	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	19.05026		
60	11/07/06	Nearest	Random	Dual	Load Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	22.70045		
61	11/07/06	Random	Random	Dual	Load Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	19.95317		
62	11/07/06	Nearest	Random	Dual	Load Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	22.70045		
63	11/07/06	Nearest	Random	Dual	UnLoad Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.97395		
64	11/07/06	Random	Random	Dual	UnLoad Point	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	19.28274		
65	11/07/06	Nearest	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	35.75745		
66	11/07/06	Random	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	32.45901		
67	11/07/06	Nearest	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	24.71455		
68	11/07/06	Random	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.88355		
69	11/07/06	Random	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.88355		
70	11/07/06	Dedicated	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.47817		
71	11/07/06	Dedicated	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	20.83871		
72	11/07/06	Dedicated	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	20.83871		
73	11/07/06	Dedicated	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.47817		
74	11/07/06	Dedicated	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	20.83871		
75	11/07/06	Class Based	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.17083		
76	11/07/06	Class Based	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	20.44515		
77	11/07/06	Class Based	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.17083		
78	11/07/06	Class Based	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	20.44515		
79	11/07/06	Nearest	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	26.37835		
80	11/07/06	Nearest	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	24.71455		
81	11/07/06	Nearest	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	24.71455		
82	11/07/06	Random	Random	Single	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	23.39493		
83	11/07/06	Random	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.88355		
84	11/07/06	Random	Random	Dual	User Defined	Multi	Simultaneous	6 x 12	4680 x 2940	3	3	100	21.88355		

A = J / (D + E)	B = (D + E) / (J / 60)		C	D	E		F	G	H	I = F + G + H	J = C x (I/100)
Av. Cycle time	**Throughput / hr**	**Comment**	**Run Time**	**No. Loads Put-Away**	**No. Loads Rertrieved**	**Bays Full at Start**	**Pre Pick-Up**	**Loaded**	**Post Deposit**	**Utilisation**	**Operation Time**
0.59257	101.25452		49.2	40	0	0	0.06399	30.80403	17.30809	48.17611	23.70265
0.67201	89.28423		49.2	40	0	0	7.67841	31.05468	15.90197	54.63506	26.88045
0.5794	103.55516		49.2	40	0	0	7.67841	26.40836	13.01903	47.1058	23.17605
0.66367	90.40656		49.2	40	0	0	7.67841	30.68829	15.5901	53.9568	26.54675
0.58821	102.00458		49.2	33	0	0	6.33469	22.82402	10.29433	39.45303	19.41089
0.61439	97.66829		49.2	40	0	0	7.67841	28.80623	13.46554	49.95018	24.57549
0.66367	90.40656		49.2	40	0	0	7.67841	30.68829	15.5901	53.9568	26.54675
0.58821	102.00458		49.2	33	0	0	6.33469	22.82402	10.29433	39.45303	19.41089
0.61439	97.66829		49.2	40	0	0	7.67841	28.80623	13.46554	49.95018	24.57549
0.61864	96.98769		49.2	22	16	0	10.33569	27.38953	10.05555	47.78077	23.50814
0.55708	107.70431		49.2	22	16	0	11.80302	27.96543	3.25811	43.02656	21.16907
0.45034	133.23281		49.2	22	16	0	8.09783	19.73932	6.94517	34.78232	17.1129
0.39597	151.52604		49.2	22	16	0	7.58573	19.73932	3.25811	30.58317	15.04692
0.65238	91.97143		49.2	22	16	0	11.08272	28.68944	10.61465	50.3868	24.79031
0.59264	101.24214		49.2	22	16	0	13.71363	28.80126	3.25811	45.7729	22.52027
0.64647	92.81226		49.2	22	16	0	11.02927	28.46638	10.43667	49.93032	24.56672
0.59264	101.24214		49.2	22	16	0	13.71363	28.80126	3.25811	45.7729	22.52027
0.55388	108.32666		49.2	22	16	0	9.21502	25.31129	8.25307	42.77937	21.04745
0.5177	115.89746		49.2	22	16	0	11.41548	25.31129	3.25811	39.98488	19.67256
0.61901	96.92866		49.2	22	16	0	10.36478	27.38953	10.05555	47.80987	23.52246
0.55708	107.70431		49.2	22	16	0	11.80302	27.96543	3.25811	43.02656	21.16907
0.65089	92.18213		49.2	22	16	0	11.02852	28.65879	10.58432	50.27164	24.73364
0.5791	103.609		49.2	22	16	0	12.68851	28.78063	3.25811	44.72726	22.00581
0.85836	69.90078		49.2	22	16	0	14.6776	36.46531	15.15315	66.29606	32.61766
0.72118	83.19736		49.2	22	16	0	16.98592	35.45661	3.25811	55.70064	27.40472
0.50167	119.60017		49.2	22	16	0	9.89125	28.85574	0	38.74699	19.06352
0.61865	96.98499		49.2	22	16	0	7.88094	29.04087	10.86029	47.7821	23.50879
0.60938	98.46066		49.2	22	16	0	8.95464	28.86228	9.24905	47.06597	23.15646
0.88066	68.1306		49.2	22	16	0	19.26564	26.9284	21.82453	68.01858	33.46514
0.33494	179.13476		49.2	22	16	0	6.13029	19.73932	0	25.86961	12.72785
0.36718	163.4098		49.2	22	16	0	3.26682	19.73932	5.35291	28.35905	13.95265
0.38236	156.91912		49.2	22	16	0	5.27964	19.73932	4.51311	29.53207	14.52978
0.61308	97.86619		49.2	22	16	0	13.25811	19.71037	14.38338	47.35186	23.29712
0.44949	133.48515		49.2	22	16	0	8.03207	19.73932	6.94517	34.71657	17.08055
0.33338	179.97699		49.2	22	16	0	6.00923	19.73932	0	25.74855	12.66829
0.35386	169.56006		49.2	22	16	0	4.89843	19.73932	2.69266	27.33041	13.44656
0.33549	178.84377		49.2	22	16	0	6.17238	19.73932	0	25.9117	12.74856
0.56392	106.39837		49.2	22	16	0	10.09627	19.73932	13.71908	43.55467	21.4289
0.39597	151.52604		49.2	22	16	0	7.58573	19.73932	3.25811	30.58317	15.04692
0.63551	94.41245		44.4	18	18	68	11.39542	30.46945	9.66292	51.52779	22.87834
0.62972	95.27989		44.4	18	18	68	11.48329	29.82459	9.75079	51.05867	22.67005
0.59681	100.53415		44.4	18	18	68	14.12812	29.82459	4.43746	48.39017	21.48524
0.59681	100.53415		44.4	18	18	68	14.12812	29.82459	4.43746	48.39017	21.48524
0.63551	94.41245		44.4	18	18	68	11.39542	30.46945	9.66292	51.52779	22.87834
0.59678	100.54032		44.4	18	18	68	13.68157	30.07557	4.63006	48.38721	21.48392
0.59678	100.54032		44.4	18	18	68	13.68157	30.07557	4.63006	48.38721	21.48392
0.54341	110.41327		44.4	18	18	68	11.94582	32.11469	0	44.06051	19.56287
0.52504	114.27655		44.4	18	18	68	12.20485	30.36613	0	42.57098	18.90152
0.66649	90.0239		44.4	18	18	68	9.87507	32.55705	11.60758	54.0397	23.99363
0.61815	97.06444		44.4	18	18	68	8.87867	30.63011	10.61117	50.11995	22.25326
0.63988	93.76796		44.4	18	18	68	10.74988	32.11469	9.01738	51.88195	23.03559
0.59566	100.72826		44.4	18	18	68	9.8625	30.30443	8.12999	48.29692	21.44383
1.09473	54.80817		44.4	18	18	68	27.48099	32.0672	29.21349	88.76168	39.41019
0.99868	60.07943		44.4	18	18	68	25.02876	29.18385	26.76127	80.97388	35.9524
0.67813	88.47883		44.4	18	18	68	12.3006	32.11469	10.56809	54.98338	24.41262
0.63551	94.41245		44.4	18	18	68	11.39542	30.46945	9.66292	51.52779	22.87834
0.54341	110.41327		44.4	18	18	68	11.94582	32.11469	0	44.06051	19.56287
0.52504	114.27655		44.4	18	18	68	12.20485	30.36613	0	42.57098	18.90152
0.56678	105.8603		44.4	18	18	68	9.97885	32.11469	3.86198	45.95552	20.40425
0.55884	107.36525		44.4	18	18	68	10.50861	30.84616	3.95659	45.31135	20.11824
0.56678	105.8603		44.4	18	18	68	9.97885	32.11469	3.86198	45.95552	20.40425
0.54118	110.86835		44.4	18	18	68	11.72708	31.65935	0.49323	43.87965	19.48257
0.51243	117.08852		44.4	18	18	68	11.18846	30.04045	0.31969	41.54861	18.44758
0.92339	64.97788		44.4	18	18	68	20.29373	31.6016	22.97422	74.86955	33.24208
0.88959	67.44697		44.4	18	18	68	19.67571	29.37535	23.07768	72.12874	32.02516
0.62122	96.58386		44.4	18	18	68	13.68522	32.11469	4.56942	50.36934	22.36398
0.59678	100.54032		44.4	18	18	68	13.68157	30.07557	4.63006	48.38721	21.48392
0.59678	100.54032		44.4	18	18	68	13.68157	30.07557	4.63006	48.38721	21.48392
0.61337	97.8204		44.4	15.33333	18	64	10.47081	27.44631	8.1316	46.04872	20.44563
0.60736	98.78887		44.4	15.33333	18	64	13.18491	27.54036	4.87202	45.59729	20.2452
0.60736	98.78887		44.4	15.33333	18	64	13.18491	27.54036	4.87202	45.59729	20.2452
0.61337	97.8204		44.4	15.33333	18	64	10.47081	27.44631	8.1316	46.04872	20.44563
0.60736	98.78887		44.4	15.33333	18	64	13.18491	27.54036	4.87202	45.59729	20.2452
0.6282	95.51164		44.4	15.33333	18	68	10.91432	27.48348	8.76404	47.16184	20.93986
0.60351	99.41787		44.4	15.33333	18	68	13.81634	27.48348	4.00899	45.3088	20.11711
0.6282	95.51164		44.4	15.33333	18	68	10.91432	27.48348	8.76404	47.16184	20.93986
0.60351	99.41787		44.4	15.33333	18	68	13.81634	27.48348	4.00899	45.3088	20.11711
0.67813	88.47883		44.4	18	18	68	12.3006	32.11469	10.56809	54.98338	24.41262
0.62122	96.58386		44.4	18	18	68	13.68522	32.11469	4.56942	50.36934	22.36398
0.62122	96.58386		44.4	18	18	68	13.68522	32.11469	4.56942	50.36934	22.36398
0.63551	94.41245		44.4	18	18	68	11.39542	30.46945	9.66292	51.52779	22.87834
0.59678	100.54032		44.4	18	18	68	13.68157	30.07557	4.63006	48.38721	21.48392
0.59678	100.54032		44.4	18	18	68	13.68157	30.07557	4.63006	48.38721	21.48392