The Use of Automatic Identification Technology to Improve Shipyard Material Handling Processes

by

Chel Stromgren

BS Naval Architecture and Marine Engineering
Webb Institute, 1991

SUBMITTED TO THE DEPARTMENT OF OCEAN ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN OCEAN SYSTEMS MANAGEMENT XIII-B
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September 2000

Signature of Author_____

Department of Ocean Engineering

Certified by_____

Professor Henry S. Marcus
NAVSEA Professor of Ship Acquisition, Thesis Supervisor, Department of Ocean Engineering

Certified by_____

Professor Nicholas Patrikalakis
Kawasaki Professor of Engineering
Chairman, Departmental Committee on Graduate Studies

The Use of Automatic Identification Technology to Improve Shipyard Material Handling Processes

by

Chel Stromgren

Submitted to the Department of Ocean Engineering
on September 1, 2000 in Partial Fulfillment of the Requirements for the Degree of Master of Science in Ocean Systems Management XIII-B

ABSTRACT

Automatic identification technology has the potential to vastly improve the manner in which large manufacturing companies, shipyards included, control material and equipment. Radio frequency identification (RFID) and other similar technologies can allow a shipyard to more accurately track, locate, and recover material and machinery within the yard, while at the same time reducing the manpower and costs dedicated to these tasks. The result of these improvements could allow future naval and commercial ships to be built more efficiently, with less material related delays, ultimately reducing the total acquisition cost.

This thesis discusses the application of RFID technology to the material control process in large shipyards. Possible improvements in material control processes are discussed in general, and the operations of the Bath Iron Works (BIW) shipyard in Bath, Maine, are examined in particular. A computer model, simulating the operations at BIW, was constructed and examined to provide quantitative data on the current practices at the yard. The model was then modified to simulate the incorporation of RFID technology into the operations. Results were compared with the baseline model and overall improvements in operating costs were calculated. This thesis presents those results and draws conclusions as to the potential benefits that RFID might offer BIW and other shipyards.

Thesis Advisor: Henry S. Marcus
NAVSEA Professor of Ship Acquisition

Table of Contents

Acknowledgements

The author wishes to thank the following for their support and contribution in making this report possible:

Dr. Henry Marcus, MIT

James Baskerville, Bath Iron Works

Brian McQue, Bath Iron Works

Chapter 1: Introduction

Automatic identification technology has the potential to significantly improve the manner in which many businesses operate. The ability to track, locate, and inventory items quickly and automatically can allow companies to increase the efficiency of their processes, reducing the effort involved in material control.

Specifically, Radio Frequency Identification (RFID) technology offers a relatively low-cost option for many companies to improve their current level of gathering and using information concerning material movement, location, and distribution. If properly implemented, this increased use of data has the potential to reduce overall costs and improve operations.

An excellent example of an industrial operation that has extensive material tracking and control requirements is a large commercial or military shipyard. Typically, for a major ship construction, the number of items moving through a yard during construction is in the millions, varying with the type and size of ship. This large volume of material requires extensive control and tracking procedures.

Since a large portion of the material and equipment that passes through a shipyard is of relatively high value, tracking requirements are often stringent, further increasing the complexity of material control. Additional complications in material tracking arise from the long lead time required for ordering many

components and the introduction of Just-In-Time (JIT) operations. For some pieces of equipment that are used in ship construction, the lead-time for ordering can be on the order of years before final delivery. In many cases, material will arrive at the shipyard long before it is needed in construction. There are cases where items may sit in storage in the yard for long periods of time before use. This situation results not only in large storage volume requirements, but also increases the chance that material will be misplaced.

In an attempt to improve construction and material-flow efficiency, many large shipyards are incorporating concepts such as just-in-time (JIT) material handling. The basic concept is that material is not brought to an assembly site until the exact time that it is needed, reducing inventory and storage requirements. Material is often stored at off-site locations and moved in small batches to the assembly site. Again, the implementation of these concepts increases the complexity of the material tracking system.

The result of all of these material tracking requirements is that a large amount of time is currently spent in the execution of these processes. Material is inventoried either visually or using bar codes at many steps during transfer and storage. Large databases are used to track material location and movement requirements. A significant amount of time is spent manually searching for lost or misplaced items.

The use of RFID in the shipyard has the potential to streamline many of these material control processes. Automatic identification technology could allow many of the most time consuming operations, such as inventory and searches to be performed quickly and automatically, reducing the burden on shipyard personnel. In addition, by improving the level and quality of information available concerning material control, overall shipyard performance can be improved. Waiting times for delinquent material can be reduced, use of available storage space can be optimized, and losses of material and equipment can be virtually eliminated.

This thesis explores the use of automatic identification technology in the shipyard environment. As part of this analysis, a material handling operation from an existing shipyard was examined. A replacement process, incorporating RFID technology, was developed. The existing and proposed processes were simulated using a computer model. The results were then compared to quantify potential gains from the use of RFID.

Chapter 2 contains an overview of RFID technology and lists some typical industrial applications. Chapter 3 presents potential uses and applications of the technology in shipyards. Chapter 4 introduces the modeling project that was undertaken to evaluate the potential benefits of RFID in the shipyard. Chapter 5 describes the software used to model the material handling processes. The current shipyard material handling process is described in detail in Chapter 6. The proposed replacement process incorporating RFID technology is described

in Chapter 7. A detailed description of the computer model is included in Chapter 8 and a comparison of the results from the simulation models is presented in Chapter 9.

Some other potential uses of Automatic Identification Technology in shipbuilding, that were not directly analyzed in this thesis are presented in Chapter 10. In addition, some potential synergies in the use of RFID between shipbuilding and on-board activities are presented in Chapter 11. Finally, conclusions concerning the use of RFID in the shipbuilding process are given in Chapter 12.

Chapter 2: Technology and Functionality

The technology involved in radio frequency identification (RFID) can still be considered to be in a developmental stage. Although there are many commercial products available and the technology has been successfully implemented in many real-world operations, new devices with improved functionality are still being developed and the overall performance of the technology is likely to improve in the future.

The basic concepts behind the technology, however, are developed and well proven. RFID is based on the concept of an autonomous tag, which can be affixed to some piece of material, and can then transmit data to a remote reader at some distance away. Information concerning the tag and/or material can be accessed at the remote reader. The type of information available, the range of operation of the system, as well as the size and cost, all depend on the type of RFID technology selected for implementation.

RFID tag technology can be divided into two basic types, active and passive. Active tags refer to radio frequency transponders powered partly or completely by a battery. Batteries may be replaceable or sealed within the device. The devices react to specific, reader produced electromagnetic fields, by delivering a modulated radio frequency response.

Passive tags are battery-free data carrying devices that react to a specific, reader produced electromagnetic field, by delivering a modulated radio frequency response. These tags have no internal power source, deriving the power they require to respond from the reader/interrogator's electromagnetic field.

Beyond the basic distinction of tag type, RFID technology can vary in the level of functionality that is provided. The most basic discriminator is in the amount of data that is contained and transmitted by the tags to the readers. The simplest, and least expensive, tags are referred to as "license-plate" tags. These devices, which may be either active or passive, simply transmit a unique identification number, which is typically pre-set in the factory during manufacturing. These tags then work in conjunction with some sort of database to store information about the material to which the tag is attached.

More complex tag systems have the ability to store and transmit data beyond a simple identification number. With these systems, the user has the ability to save information on a particular tag and then to read this information remotely. Actual information about the material in question can be recorded directly on the tag, eliminating the need to cross reference a database. This can speed up the identification process and allow a user to work independently of a central computer system. Typically, this functionality is found only with active tags.

Depending on the complexity of the tags and the reading system, RFID can provide additional functionality in the locating of material. Several types of active tag/reader combinations have the ability to determine the range that a tag is being read at. This gives the user the ability to discriminate between tags that are being read in a large area. Additionally, if multiple fixed readers are being used, the range finding ability of a tag can allow users to triangulate the exact location of a tag.

RFID technology has been implemented in many industrial settings in various configurations using different sets of the functionalities identified above. The basic types of RFID implementations can be divided in to five distinct categories:

Information/Identification/Inventory: The most basic usage of an RFID system is to provide a method for storage and retrieval of information about material. This type of operation can be used in many instances in an industrial setting.

Most simply, the technology can be used to identify individual pieces of material within a system. Workers using hand held scanners can interrogate tags to identify material and search for particular items. Similarly, tags and readers can be used for inventory purposes, quickly scanning a large number of items. The system can also be used to store relevant information about a piece of material.

History, delivery deadlines, or storage requirements could all be stored on a tag and instantly recalled for use.

Tracking Within Defined Paths: Highly-automated assembly line processes can use RFID tags to track progress through a system. In these cases, fixed readers are positioned throughout a system and automatically scan and record tag data as it passes. Information is sent to a central computer system which identifies material and current location. This type of system works well to track large volumes of material passing through fixed lanes of movement.

High-Volume Sorting: RFID tags can be used to dramatically improve the performance of high-volume sorting operations. In applications where large amounts of material are sorted based on some identifying characteristic, RFID can be used to speed up or automate the process. For example, in laundry applications where items are processed in large batches and then sorted back into groups for individual owners, the distribution process can be extremely laborious, involving manually reading individual labels or nametags. With small RFID tags attached to the items, the process is greatly improved. Distribution information can be automatically read and sorting could be largely automated based on this data.

Location Finding: A more advanced usage of an RFID system is to search for the location of particular object. There are various methods in which tags can be

employed to provide this functionality. Fixed tag readers can be distributed throughout the system in all storage locations. The readers can then be queried to search for a particular tag. When a tag is registered, the tag location can be determined to be within the read area of a particular reader.

More complex RFID tags and readers can actually be used to triangulate a tag's position in a particular setting. If tags are used that have the ability to determine range, multiple readers can be used to mathematically triangulate the exact position of a tag. This function is extremely useful in warehousing situations, in which large amounts of material are stored in large areas. The triangulation function is often combined with electronic maps or schematics to visually show the location of material.

Condition Monitoring: RFID tags have been used in some applications in combination with remote sensors to monitor the condition of a piece of equipment. Small sensors that can register variables such as temperature, pressure, humidity, etc., can be electronically linked to the RFID tag. When queried, the tag can then transmit this data to a reader. Operators of the system can use this information to monitor the condition of the equipment. A system such as this can also be set-up to automatically report if conditions fall outside a certain range. This is particularly useful in material control situations in which environmental conditions are a factor in storage.

These five basic types of operation are used in various combinations in different industrial applications, such as; warehousing, production, service, and traffic control. For example, in a warehousing situation, several types of RFID operation might be used to improve efficiency. Fixed scanners could be used at gates to identify and inventory incoming and outgoing material. Multiple fixed scanners could be employed within the warehousing area to search for particular items and triangulate locations. Hand-held scanners could also be used by operators to locally search for and identify material.

The optimal set-up for each application depends heavily on the volume of material being processed, the amount of manual work required in the present processes, and the types of activities being performed. Chapter 3 will explore some of the potential applications of RFID to the shipyard environment.

Chapter 3: Potential Applications in the Shipyard Environment

While any of the basic RFID operations may be used on a limited scale in the shipyard environment, there is potential for wide-spread usage of three types of RFID operations. Because of the high volume of material being processed, the large storage requirements, and the strict scheduling of arrivals and delivery within the yard, automated inventory, location finding, and material flow monitoring may all be used effectively within a yard to lower material control costs.

While each of the major shipyards practice their own material movement and control procedures, the general movement of material is similar at every yard. Typically, material arrives at a yard via truck, rail, or barge at an incoming warehousing facility. This facility is sometimes located within the yard itself or sometimes is an off-site location. As material arrives, it is inventoried, entered into a material database and tagged with the yard's own tracking information (most often ID numbers or bar codes). Some material and equipment is then grouped together based on location of use, system, or required delivery time. These groupings are generally packaged together in some manner, either on pallets, or in storage boxes and the groupings and pallet/box number also recorded in a central database. Additional data concerning each part can also be included in the material database, such as required delivery time and location, assembly unit to which it belongs, and testing/delivery instructions.

Packaged pallets and boxes, as well as large individual items are then dispatched to various storage locations throughout the yard. Locations are usually predetermined, based on the site where the items will be needed, when they are required, and whether items must remain indoors or can stand weather. Upon delivery of parts to locations in the yard, all items are inventoried and the database updated with location.

Most major shipyards have implemented some type of Just-In-Time (JIT) delivery system for material and equipment within the yard. Under these sort of schemes material is identified in a database with the location and date that it is needed in the assembly process. The advantage of JIT is that assembly workers do not have to go searching for material as it is needed in the construction process; rather the material is delivered as needed, and on-time. This system also eliminates the need to store material at the actual assembly site, where space is typically restricted. The major difficulty with JIT involves the control of material. Every operation in the construction of the ship has to be carefully scheduled before hand. All material must be delivered as promised or the entire assembly process can be thrown off, wasting time and resources.

Within a JIT system, each part or piece of equipment is assigned a delivery time and location within the database. The equipment stays at its in-yard storage site until the proscribed delivery time. At that point yard workers locate the material, based on its identified storage location in the database. Typically, entire pallets

or boxes will be loaded with material that has the same delivery time/location. After the proper pallet/box is located the material is again inventoried. It is then transferred either to the location where a particular sub-assembly is being built or to the platten for placement on-board the ship.

When it works correctly, this type of material control system accurately tracks the location and status of all items in the yard. Storage locations, groupings, and required delivery data are recorded and stored for all material. However, because of the manual way in which data is collected, the operations tend to be time intensive. Multiple manual inventories and part identifications must be performed. In addition, the manual nature of the system allows for errors to occur in identification, data entry, and part location. These errors are especially troublesome because they can lead to large amounts of time involved in searching for material and in delays in assembly.

In order to help quantify the potential advantages of the use of RFID in the shipyard, an effort was undertaken to simulate the shipyard processes using a computer model. This modeling project is described in Chapter 4.

Chapter 4: Description of Modeling Project

While the shipyard environment seems like an ideal location to implement RFID technology, it is difficult to determine the exact improvements that could be delivered by such a system. While the time spent performing routine events, such as inventory or delivery, is fairly easy to quantify, time spent on extra ordinary events, such as searching or waiting for material, is often not recorded and is more difficult to measure. Additional costs due to lost material and downstream time lost due to delinquent delivery also adds to the cost of material control.

In order to try and quantify what the potential gain is to a typical shipyard, it was decided to perform a computer simulation of yard activities. This simulation would be completed both with the current material control system and with an RFID based system. The results would then be compared to quantify the gains achieved by the new system.

Bath Iron Works (BIW), a private shipyard, located in Bath, Maine, builds primarily large, complex ships for the U.S. Navy. The shipyard has recently invested in an internal research project to examine the potential for the use of RFID on the shipyard. As part of this effort, BIW surveyed available RFID technology and contracted with a supplier to develop new advanced tags for testing in the yard. The tags that were developed during this research effort were

active tags that had only "license plate" data transfer capability, but which had the ability to be used for range-finding and triangulation.

BIW agreed to work with MIT to study their current material control practices and to develop a notional implementation of the RFID technology. The current BIW processes and the proposed RFID processes would then be modeled at MIT using a computer simulation to predict gains.

The first step in the modeling process was to survey BIW's current operational practices. Typical material movement operations were observed over a period of days. Storage and processing locations were charted, information storage, retrieval, and transfer, were documented. Typical material volumes and distributions were observed. The current in-yard processes are described in Chapter 6 of this document.

After completing the in-yard surveys, the current practices were modeled on the computer using a tool called ProModel. The operation of this software is described in Chapter 5. The model developed captured typical material movements into and through the shipyard. Events such as inventories, retrievals, and searches were all modeled. Individual times for material processing were monitored for each item and total processing times calculated for blocks of material. Loss of material was also simulated and the total number of items lost calculated.

All the of the results from this baseline model were summarized and compared, when possible, to the actual performance experienced within the yard. Actual quantitative observations of full-scale performance in terms of man-hours consumed and items lost are hard to obtain. Results are compared to full-scale on a mostly qualitative basis. In cases where results did not match, the accuracy of the modeling or of the input variables was checked and the model was modified to reflect reality.

As part of this verification effort, a sensitivity analysis was also performed on the model. In this effort, input variables were systematically varied and the results of the model monitored. In cases where small changes in input yielded large changes in output, the accuracy of the variables was double checked. Careful attention was placed on these variables during subsequent operation of the model to insure that results would be reasonable.

Once a robust model had been developed and verified against real-life performance, the model was modified to include the proposed RFID implementation. The actual RFID proposal developed by BIW and MIT is described in Chapter 7.

The RFID model was then run with the exact same input variables as were used in the baseline model. Again, processing times and lost material volume were monitored. Results were then summarized and compared with the baseline

model. Improvements and degradations in key performance parameters were noted and an overall prediction of improvements due to RFID was made. The results of the model comparison are presented in Chapter 9.

Chapter 5: Discrete Event Simulation

Discrete event simulation is a tool that has recently been used by many manufacturing companies to analyze and optimize their production processes. These computer programs allow users to model complex processes on the computer and to then modify them and experiment in order to optimize overall performance.

The particular tool used in this study is ProModel, a discrete event program written and distributed by the ProModel corporation. The software has become one of the standard programs used by the manufacturing industry to analyze production processes.

Discrete event simulation tools are extremely effective because they have the capability to simulate complex systems that involve interactions between multiple processes. These interactions can often be difficult to analyze manually and are often counter-intuitive in their operation. Programs such as ProModel analyze the operation of each individual process (each discrete event) and then predict the interactions between them and the operation of the overall system.

In building a model, the user is first required to define the overall layout of the system and is then required to describe the logic of operation for each individual event. The physical layout of the system is defined, including locations of events,

paths of movement, description of material, and identification of personnel operating the system. Variables are described, such as the rate at which material enters the system and the down times of particular resources.

In defining the logic for each discrete event, the user must capture the actual operation of the real-life resources. Factors described in the logic include: material used during the event, time required to process material, personnel required to complete the process, man-hours required, the output products of the process, and the destination of material once complete. Equally important, the user must describe the logic used in making decisions at each step. How the following decisions are made must all be carefully defined: the order in which items are processed, the use of particular workers, the distribution of products, and the variability in processing. It is the interactions between the logic for each individual step that the program uses to determine the operation of the overall system.

Typically, the first step in creating a process model is to survey the actual operation of the current system on-site. All the factors that must be used to define the physical layout and logic description are observed and recorded. Obviously, in some cases, it can be difficult to obtain exact data on the system operation. In these cases, best estimates must be made, based on observation and experience. Data obtained in this stage concerns the operation of each individual element of the system.

Once the data has been obtained from the actual process, an initial model can be built. The physical layout and the logic can be programmed into the simulation, along with the estimates for the variables used in operation. The simulation is run and results calculated for the overall operation of the system.

Once the initial results are produced and evaluated, they are then compared to the operation of the complete real-life system. Data is obtained from the actual system on overall performance and operation. The results of the model are analyzed in comparison to the real-life data to find deviations. Once any irregularities are found the logic, layout, and variables in the model are analyzed to determine the source of the errors. This process can be time-consuming and may require many iterations before the performance of the simulation model corresponds to the real-life system. The process of fine-tuning the model is extremely important, however. In order to perform analysis of changes to the system and to optimize operation, the performance of the current processes must be accurately modeled.

In addition to checking the performance of the simulation to the actual system, a sensitivity analysis is usually also performed of the simulation model. In this analysis, input variables to the system are individually varied in small increments over a reasonable range and the model rerun for each change. The outputs for each run are analyzed to determine the effect of the changes in the variables on

the performance of the overall system. This type of analysis determines which variables have the greatest impact on performance and if there are certain ranges in which variables have extreme effects on results. This data helps in the running of the model and in the future optimization of processes.

Once a robust model is completed, experiments can begin on the simulation. Changes in layout, manning, variables, and decision logic can be included in the model and the overall performance calculated. These results can then be compared to the baseline model to determine the effect on the operation of the system.

If an optimization of a particular process is to be performed, multiple runs must be completed with varying modifications to the model. During these runs, changes to the model must be made in an organized, incremental manner. Results must then be examined to determine the effect and degree of change to the model results. Additional changes, based on the results can then be made until an optimal solution is found.

Chapter 6: Description of Current Processes

The material control processes used at BIW are similar to the typical shipyard practices described in Chapter 4. In general, the operations at BIW are relatively advanced in the level of technology used and information collected during the material control process. The yard utilizes an extensive material database to track and store information. In addition, the yard has fully implemented JIT practices throughout the construction process. In doing so, they have developed a sophisticated, distributed system for receiving, warehousing, storage, and distribution of material.

Receiving: All material used for ship construction by BIW is received at one of two off-site facilities, located a few miles from the yard. All structural material, steel plate, shapes, etc. is delivered by truck or rail to the Harding Plant. All other non-structural material and equipment is delivered to the East Brunswick Fabrication Facility, also located a few miles from the yard.

Upon arrival at the Harding Plant all material is inventoried and tagged with BIW ID numbers and bar-codes. ID numbers are assigned from a material control plan developed in advance by planners. The material is then placed in storage at the off-site facility until needed. Because much of the material is generic, that is, specific pieces are not assigned to specific uses, but are used as a bulk

commodity, the material control procedures are not especially stringent at this point. Material is sorted by type and size and stored in lots.

The steel fabrication shops are all located within the Harding Plant. Cutting, blast and coat, and sub-assembly activities are performed directly at the site. Structural parts are cut from bulk material and coated. Some parts are welded together into sub-assemblies. Once cut or assembled, all parts are given a specific identification number, which is assigned from the parts database. Items are then grouped together based upon the assembly or system of the ship that they are part of and based on the time that they are required in the yard. Groupings are placed together on a pallet or in large metal storage boxes. Part groupings are pre-determined by planners and specified in a material control plan. As the parts are grouped together, their ID numbers are recorded and stored in the material database. The pallets and boxes, as well as large individual structural items are then ready for shipment to the main yard.

Parts are shipped from the Harding Plant to the main yard in two manners. Most material is placed on large flat-bed trailers. These trailers are parked, without cabs, in the facility and loaded as parts are ready. When the trailers are full, they are driven by cab to the main yard. Some material, which is needed quickly or has special storage requirements, is place directly on trucks for immediate transfer to the yard.

Non-structural material and equipment arriving at the East Brunswick Facility is handled with a process similar to that used at Harding. Upon arrival at East Brunswick, material which is unique (material which is specifically destined for a particular application) is given a identification number and is bar-coded and stored. Generic material, such as pipe and fittings, are inventoried, coded, and stored in bulk.

Material is stored at the facility until needed and then grouped according to usage. Again, groups are placed on pallets and in metal boxes and the material groupings recorded in the central database. All material is then delivered to the main yard by truck.

Delivery to Yard and Storage: Upon arrival at the main yard, all vehicles, flat-bed trailers and trucks, pass through the main delivery gate. Vehicles stop at the gate and the driver must perform a manual inventory of all items on board. The inventory, at this point, records pallets, boxes, and individually shipped large items. This inventory is performed manually using bar-code readers hooked to the main database. Once the inventory has been completed, the trucks and flat-bed trailers proceed into the yard. Flat-bed trailers are parked, detached from the cab, and left in the yard. Trucks proceed around the yard, unloading pallets, boxes, and material at various storage locations.

Storage locations for all material are predetermined and assigned through the material database. Planners select material storage locations based on where material will be used and availability of storage space. Drivers deliver trailers and material to the pre-planned storage locations. If the location is full, or if the driver must store material at an alternate location for some other reason, he records the new location in the material database, so that it can be found at a later time.

Material on trucks is unloaded directly at the storage sites and the trucks proceed back out of the yard. Flat-bed trailers are parked and detached from the cabs. Material is unloaded from the trailers as needed and the trailers remain in the yard until they are empty. In some cases, if just a few pieces of material remain on a trailer, the items will be moved to ground storage locations, to free up the trailer. These moves are also recorded in the database.

Material Recovery: All stored material, except bulk items, are assigned times and locations that they are needed in the shops or on-board a ship for construction or installation. This data is recorded, along with the storage location, in the material database. When the required usage time approaches, waterfront personnel locate the material and deliver it to the required destination. In locating an item, a worker will check the database for the assigned location; he will then proceed to that area and search manually for the particular item. This is

done either by visually checking tags or by scanning bar-codes. Once an item or pallet is found, it is transferred to its destination by truck or by forklift.

Problems: This material control plan functions well and adequately tracks most material throughout the construction process. However, there are several inherent problems with the current system that reduce the shipyard's overall efficiency in handling material. The most significant of these problems is the large amount of labor that is consumed in identification and inventory of items. As pallets, boxes, and items are moved, stored, and recovered, ID numbers must be manually checked and recorded. Inventories, which take place as material enters the yard, are extremely time consuming. The result is that yard workers spend a large deal of time reading tags or bar codes and entering data. In addition, these processes also have large potential for error. Mistakes in reading or entering data can lead to errors in the material database.

Another significant problem with the material control system is in the loss or misplacement of items. This type of loss can occur in many different manners. If tags are misread and data is mis-entered, improper storage locations can be recorded in the database. In addition, if drivers forget to record actual storage location when they cannot place items in the pre-designated location due to inadequate capacity or misplace items due to simple error, material can no longer easily be located. Additionally, if material is transferred within the yard, to

make space, or to clear off a trailer, often the moves are not recorded in the database.

In cases where the actual material location is not recorded in the database, the process of recovering items becomes much more difficult. Waterfront personnel will proceed to the intended location and begin a search for material. Items at the location will be checked, looking for the intended item. When it is not found, workers must then begin searching adjacent areas for the material. Depending on how far away from the intended location the item actually is and the quantity of items that must be searched among, the location of the proper material can take hours or days.

This search for material is troublesome not only because the search takes many man-hours, but also because, with the JIT process, the material is needed in the construction process immediately. When material does not arrive on time, construction work is delayed and man-hours are wasted waiting for material.

In some cases, where material is not found in a reasonable amount of time, the yard is forced to order replacements, which wastes times and money. In other cases, yard personnel use identical or similar items as replacements. If these replacements are not properly recorded, other jobs can be delayed as the replacement material is later searched for.

Chapter 7: Description of Proposed RFID Process

RFID would appear to offer some distinct advantages to BIW because its functionality directly addresses the problems with the material control system outlined in the previous chapter. The features that such a system could offer would be improved speed and accuracy in identification and inventory, as well as improved capability for location finding.

After surveying the current material control procedures, BIW and MIT worked together to develop a notional implementation for an RFID system at the shipyard. This plan was based on the tag technology developed by BIW under its RFID research project. The tags developed under this effort were active license plate tags, with long-battery life, that also had capability for range finding and triangulation. In testing, these tags were demonstrated to have long read ranges and fast read times. The system also demonstrated an ability to scan a large number of tags simultaneously within the range of an individual reader.

The tags were built with an internal metal backing plate within the casing. The tag antenna was then tuned to account for this plate. This tuning allows the tags to be directly affixed to large metal objects without degradation in performance. Since the antenna is already tuned to account for metal backing, the presence of metal behind the tag does not affect the quality of the transmission.

The proposed RFID system consists of tags and multiple readers. Tags are permanently attached to all pallets and metal storage boxes used for material movement in the yard. In addition, there is a pool of available tags that can be attached to individual large items for identification. Each tag is coded with a unique identification number that is transmitted when the tag is scanned.

The reading system for the yard consists of three separate components; portable readers that can be used by personnel to individually scan items, fixed readers at yard gates to inventory items as they pass into the yard, and fixed readers on towers, distributed throughout the yard to triangulate the location of material.

The modified material control plan is essentially similar to the current process, but the use of the RFID tags has been included to streamline identification and location. The process begins in the Harding and East Brunswick facilities as material is prepared for shipment to the main yard. Before items are grouped and placed on pallets or in boxes, the attached RFID tag is scanned using a portable reader. Then, as the items are placed on the pallet or box, their bar codes are scanned and associated with the tag ID. In the material database, all items are then associated with a particular RFID tag. A complete inventory for each pallet or box is listed against the ID number.

Similarly, large individual items have their part number associated with an RFID tag number. As the tag is attached to the part, both the tag and the bar codes are scanned. This data is stored in the material database.

Material is then transported to the yard as in the current system. As the material passes through the main gate, however, fixed readers query all tags and create an inventory of all the tags on the truck. The database then determines what material is associated with those tag numbers and records that material as having entered the yard. The process happens very quickly as the truck slows down to enter the yard. the driven no longer has to exit the cab, climb on the bed, and manually inventory and record the items on-board.

Once in the yard, the driver can then use a hand-held scanner to determine the parking location for trailers, or the delivery location for pallets, boxes, or individual items that are to be unloaded. The driver would have the ability to scan all of the items on a truck and cross reference the database to determine delivery points. The driver would then be presented with a list of delivery locations and the items to be distributed at each. Material could then be more quickly delivered, unloaded and stored.

In recovering items for delivery to shops or to platten, both the fixed readers, the towers, and the hand-held scanners are used to improve the identification process. When material is needed for a particular job, the yard worker

responsible for collecting material checks the database. The system identifies all of the items that will be required. It then cross references all the items to the pallets or boxes in which they are located, if applicable, and to the RFID tag number. The system then queries those particular tags via the fixed readers on towers throughout the yard. The tags are identified and triangulated to determine the exact position in the yard. This information can then be displayed on an electronic yard map, showing the worker the storage locations of each item. The system being developed by BIW can determine the location of an individual tag to within an accuracy of approximately 10 feet.

As the worker proceeds to each storage location, he can then use a hand-held scanner to find the actual pallet or box on which the material is located. By quickly scanning items in the storage area he can find the desired item without having to inspect each item and manually read labels or bar codes.

This system would eliminate problems due to material being misplaced or lost in the storage process. Material mistakenly stored in wrong locations or shifted during the storage period can easily be located and recovered.

The system installed at BIW would also be used to track the progress of certain material as it moved through the construction process. The tags can be used to monitor the delivery of equipment and machinery into the yard, to the assembly sites, and finally onto the ship. Tags affixed to the equipment can be scanned as

it is delivered and installed using hand-held scanners. Production control engineers can then monitor the progress of equipment through the yard and can make adjustments before material becomes delinquent.

This type of material monitoring gives the construction planners better visibility of the progress being made in construction and outfitting. By monitoring the progress of machinery and equipment into units and onto the ship, a picture of the overall progress can be maintained. In addition, with better visibility of material movement, the planners are more able to respond to problems and delays in construction. Alternate plans can be developed in time to forestall major delays in the construction process.

Chapter 8: Description of Process Model

The process model developed by MIT simulates the movement of material and equipment through the main BIW yard, beginning as it enters the main gate and ending as it is delivered to the platten or shops for use. The model simulates the arrival of material by truck, the inventory of incoming items, and the delivery to storage locations throughout the yard. It then simulates the storage and recovery of all material, including the misplacement of items and the subsequent searches by shipyard personnel. Finally, the model simulates the recovery and final delivery of all items.

The model is run in two separate formats. First, the baseline model is run, representing the current practices used at BIW. Then, the model is rerun with some modifications to represent the system after the incorporation of RFID technology into the process. First, the baseline model will be described.

The model is based on a simplified physical layout of the actual shipyard. Actual locations of gates, roadways, buildings, and plattens are modeled. Paths of movement for trucks and personnel are also accurately modeled. Parking locations for flat-bed trailers in the yard are identified and are similar to what are used in the BIW yard.

The locations of storage areas for material is somewhat simplified. In the actual shipyard environment over 100 actual storage locations are individually identified and assigned. To simplify the operation of the model, storage areas have been grouped into 20 locations, based on their proximity to one another. Adjacent storage areas have been grouped together. Storage capacity and search times are all calculated based upon those that would be experienced for the individual areas, so that the final results of the modeling will not be affected by the simplification.

A layout of the model is shown in Exhibit 1. In this layout, a representation of the BIW shipyard is shown in a schematic view. Buildings and areas with major obstructions are shown as shaded blocks. Ground storage locations and flat-bed trailer parking locations are indicated on the plan. Areas labeled S1 – S20 represent the designated ground storage locations. Areas labeled F1 – F16 represent the flat-bed parking spots. In order to facilitate on-screen display and printing, this schematic of the yard has been foreshortened in the transverse direction. Actual distances between locations and travel times, however, were calculated using the real dimensions of the yard. The compressed schematic is for display purposes only and does not affect the results of the model.

There are two distinct processes modeled in this simulation; the delivery of individual items by truck and the delivery of items by flat-bed trailer. These two

processes are linked only by their endpoint, the delivery of items to the waterfront.

In the first process, items are brought into the yard by truck, they are inventoried and then delivered to the prescribed storage locations and unloaded. Items remain at those locations until they are needed. Material is then recovered by shipyard personnel and delivered to the location where it is needed.

In the second process, items are loaded onto flat-bed trailers, which are driven into the yard by truck cabs. The items on the trailers are inventoried and the entire trailer is then driven to a parking location and stored. Items are recovered directly from the trailer as needed and delivered. When all items have been removed from the trailer, the cab returns and removes the flat-bed. Occasionally, a few remaining items will be shifted from the flat-bed trailers to ground storage to free up the flat-bed and the parking place.

All material in this model is treated as identical commodities. In the actual yard, material that is processed could include; collections of individual steel pieces grouped on a pallet, individual or multiple machinery items on a pallet, or large, individual steel assemblies. However, since each of these types would be identified by an individual RFID tag and would be recovered and delivered as a whole, they are treated as a single unit. Loading and unloading times, as well as storage space required, that are used in the model, are based on an average

time for all items, so the physical characteristics of individual items are not relevant. Therefore, all items are treated as a single type in the model.

All items entering the model are assigned an intended storage location and a required delivery time for use. These values are assigned from an external array, which is written by the user. This is similar to how the actual process operates, when storage locations and times are pre-determined by material control personnel. Assigned storage location and storage time are stored as permanent attributes for each entity. This data travels with the entity as it moves through the system.

In the first process, items are loaded onto another entity, designated as a truck in the model. A variable number of items are loaded onto each truck before it enters the yard. The number of items is randomly generated by the program from a standard distribution of values.

As items are loaded onto the truck, the actual delivery location of each item is determined by the program. In most cases the actual delivery location is the same as the assigned delivery location taken from the input array. However, in cases where the assigned delivery location is already full, the item is given a different actual delivery location. The alternate location is the next adjacent storage space that has adequate capacity. In addition, in a certain percentage of

cases, an alternate, actual delivery location is assigned at random. This is done to simulate cases in which a driver drops off items at the wrong storage location.

Once loaded, the truck enters the yard at the main gate. The truck stops and all items are inventoried at this point. The time taken to complete the inventory is based upon the number of items on the truck and a statistical distribution of inventory times.

The truck then proceeds on a set path through the shipyard, passing by all storage locations. At each location where there is an item or items to be delivered, the truck stops and unloads the items. The unload time is based on the number of items and a statistical distribution of unload times. Once all items have been unloaded from the truck, the vehicle proceeds back to the yard gate and exits.

All items are unloaded at the actual delivery locations determined by the program. In order to simulate the search process for misplaced items, placeholders are used to indicate when the actual delivery location of an item is different from the assigned delivery location. In these cases, when an item is unloaded from a truck at the incorrect location, blank placeholders are instantly delivered to each storage location between the actual location and the assigned location. Placeholders are assigned a storage time identical to that of the item

that they represent. These placeholders are used later, in the recovery process, to simulate a search for the item.

All items remain in their storage locations until their assigned storage periods are completed. At that point, a yard worker is called to the storage location to recover the item. Once at the storage area, the yard worker takes some amount of time to find and recover the appropriate item. This time is calculated based on the number of items currently in the storage area and a statistical distribution of search and recovery times.

In the cases in which the actual and assigned storage locations are not the same, the recovery procedure is more complex. Before being called to the actual storage location by an item, the yard worker will first be called to the assigned location and any intermediate locations by the blank placeholders that were delivered. The worker proceeds to each of these areas and removes the placeholder. The time taken to remove each placeholder simulates the time to search that area for the specified item and is again based on the number of items at the location and a statistical distribution of search times. After recovering all of the placeholders, the yard worker then proceeds to the actual storage location and recovers the item. Once recovered, items are delivered to the waterfront, where they exit the system.

The second process, simulating delivery on flat-bed trailers, proceeds in a manner very similar to the first, except that all items in an individual trailer load are delivered to a single location. As items are loaded onto a flat-bed trailer, they are again assigned a specified storage location and a specified storage time. All items in a particular load have the same specified storage location. Storage times, however, can vary for each item. The number of items in a particular load is again variable and determined by a standard mathematical distribution.

The process for determining actual storage locations is similar to that for the first process. In this case, however, an alternate delivery location is assigned if there are any items in the assigned delivery location. This indicates that the flat-bed trailer on which items are delivered and stored is still parked at the storage location. Therefore, another trailer cannot be delivered to the assigned spot. The program then finds the nearest empty location and assigns it as the actual storage location. Again, in a certain number of cases an alternate actual delivery location is given, even when the assigned location is empty, to simulate the misplacement of items. In this process, all items on the truckload are given the same actual storage location, since they are delivered together on a flat-bed trailer.

After being loaded, the trailer is taken by a truck cab to the main yard gate. As in the first process, the truck stops at the gate and all items on-board are

inventoried. The truck then proceeds directly to the actual storage location. The trailer is then disconnected and the cab exits the yard. The delivery time, in this case, is not dependent on the number of items in the group.

Again, in cases where the actual and assigned delivery locations do not match, blank placeholders are used to indicate locations that must be searched. In the flat-bed process, however, since all items in a particular delivery have the same assigned and actual locations, multiple placeholders are placed at each storage spot, one for each item on the trailer. These are delivered at the time when the trailer is disconnected from the cab at the actual storage location. Again, individual storage times, matching those of the items on the trailer, are assigned to the blank placeholders.

Items are recovered from the trailers as their assigned storage periods are reached. Yard workers are called to the trailer, where they search for the item. Search times are dependent on the current volume of items on the trailer and on a statistical distribution of search times. Once the desired item has been recovered, it is unloaded from the trailer and delivered to the waterfront.

As in the first process, if a trailer has been parked in the wrong location, the yard worker must first recover all of the blank placeholders before proceeding to the actual storage area to recover the item. The worker is called by the placeholders to each location between the assigned and actual locations. The worker

searches each area for a period of time based on the number of items in the location. The worker then removes the placeholder and proceeds to the next location.

In some cases, determined by a mathematical distribution, when only a few items remain on a trailer, remaining pieces of material are transferred to a ground storage location. In this case, the assigned storage location is changed and a new actual location is assigned, following the procedure described for the truck delivery process. Once all items have been removed from a trailer, a truck cab returns to the parking location. The trailer is attached and driven back out of the yard.

For both the truck delivery and flat-bed trailer delivery processes, all processing times are monitored by the software. During the delivery and recovery processes, time spent inventorying incoming items, delivering material, searching storage areas for intended items, searching for lost items, and recovering material are collected. In addition, the number of misplaced items, the number of transferred items, and the total number of items passing through the system are measured. This data is collected by the software and presented to the operator to examine the overall performance of the system.

After completing the baseline runs, simulating the current operation process, the model was modified to simulate the implementation of the RFID technology. This

step was done in two stages, in order to analyze the benefits of each type of RFID equipment. By analyzing the implementation in stages, the relative cost reduction associated with each type of technology could be identified. This would enable the yard to make better decisions over how to deploy this technology in the future.

First, the simulation was run with the implementation of fixed scanners at the gate to perform inventory and hand-held scanners, operated by yard personnel, to improve area searches and material identification. There were several modifications that were made to the simulation model to represent the modified process incorporating this RFID technology. Most of the changes involve the modification of the statistical time distributions for the events that occur in the process. Time spent in inventorying incoming items and searching storage areas for particular items are both reduced. This reduction represents the operator's ability to use fixed and hand-held scanners to quickly identify material. In addition, loading/unloading times for trucks and trailers are also reduced. These reductions are due to the improved speed with which workers and drivers can scan material and enter data into the material control system.

The model was then further modified to include the implementation of fixed tower based scanners and triangulation to locate lost material. The major change in the RFID model to incorporate the triangulation technology was the elimination of the blank placeholders that represent the search process for missing items. In

this model, items are still misplaced when storage location are full or when a driver makes an error in delivery. Now, however, because actual storage locations can be determined before recovery, using the tower-based fixed readers, yard workers can proceed directly to the actual storage location, without having to search any intermediate areas. With these changes the model now simulated the full implementation of the RFID plan developed by BIW and MIT.

Once the changes had been made to the models, both were run using the exact same input data and conditions as in the baseline case. Exhibit 2 describes the variables and input parameters used to run each model. The model was run with a total simulation period of approximately 200 working hours. The entry of 1000 ground storage units and 500 flat-bed storage units was simulated.

During each run, the same time and material factors were monitored and examined. The two runs were then compared to determine what improvements were attributable to the introduction of the RFID technology.

It is difficult to estimate the exact percentage of material that is misplaced, mislabeled, or lost in a given period in the shipyard. In fact, the volume of misplaced material will actually vary, depending on a number of factors, including; the workload placed on personnel, the level of urgency that exists on a given project, and other factors, such as weather, which make identification and handling more difficult.

In order to examine the benefits of RFID implementation over a range of possible material misplacement rates, multiple runs were completed using the baseline simulation model. The overall rate of material misplacement is a sum of two different amounts, the level of material that is accidentally misdelivered by a driver, and the level of material which is misplaced because the intended location is full at the time of delivery. The first factor is specified as an input to the model. The second factor is determined by the software as a result of the simulation. In order to vary the overall rate of material misplacement, the percentage of material mishandled by drivers was varied. The input value was iterated until the desired overall misplacement value was reached. The simulation was run with an overall rate of 5% to 30%, in increments of 5%. The results of the model runs are described in Chapter 9.

Chapter 9: Model Results

Using a given set of input parameters, each run of the simulation model will produce somewhat different results, due to the variable nature of some parameters in the program. Values such as, search times, material misplacements, and material flow rate, are generated randomly, within defined parameters, in the simulation. Because of the variable nature of these factors, the overall results produced by the model can vary somewhat, using a given set of inputs. To account for this variability in results, multiple model runs were completed at each increment of material misplacement. Ten separate runs were performed and data sets for each increment were averaged to produce a typical set of results.

Multiple runs were also performed and averaged for the RFID models, using the same input parameters as in the baseline. For the first RFID model, runs were completed at the same increments of lost material as for the baseline. For the second RFID model, in which no material is considered to be lost, only a single set of data was produced.

Exhibit 3 shows the overall averaged results for the baseline model runs. Data is presented individually for both material which is stored on flat-bed trucks and for material that is unloaded into storage areas in the yard. Combined average data for all material in the yard is also presented.

For each type of material, several sets of data are provided. The initial group shows the overall time spent on various operations in the simulation. The first row shows the total time spent in the process of conducting the inventories on items being delivered on truck through the yard gate. The second row shows the total time spent searching local areas in the delivery of parts. This represents the time that yard workers spend searching through the actual storage location for parts, it does not include time spent searching in incorrect locations. That value is given in the next row, remote searching. This number represents the total time spent by yard personnel searching for misplaced items in incorrect areas.

The next set of data for each type shows the same data as in the first section, but averaged for individual items. The first row in the second section shows the average time spent on inventory per item. The second row shows the average time spent on local search per item. The third row shows the time spent searching remote areas averaged for all items in the system. The fourth row shows the remote search time averaged only for misplaced items.

The last section for each material category details the number of items misplaced and the average time material took to be delivered. The first row lists the total number of items of that type that were misplaced in the simulation. This number is out of 1000 ground storage units and 500 flat-bed storage units, for a total of 1500 units passing through the system in this simulation. The second and third

rows both show the average time that elapsed in the simulation between when a part was requested for delivery and when it was found and delivered to the proper location for installation. The second row shows the average delivery time for all units in the system. The third row shows the average delivery time just for items that had been misplaced.

In looking at the data in Exhibit 3, it can be seen that the level of misplaced material has a very strong effect on the overall results of the model. Exhibit 4 shows average time spent searching for misplaced items, as a function of misplacement percentage. This plot shows that the remote search time actually increases greater than linearly as the misplacement level rises. This is due to a number of factors in the operation of the yard. As more and more material is misplaced, yard personnel spend a greater amount of time having to search for items. As a result, the volume of items in storage locations increases as material is recovered more slowly. This increased volume further adds to the amount of time workers spend in the search process. In addition, as search times increase, storage locations fill up more often. Additional items will be misplaced as they are put in alternate locations, again compounding the delivery problems.

Another factor that leads to the rapidly increasing search times, is the tendency for searches to be abandoned and replacement parts improperly taken as replacements. As search times increase, yard workers will eventually stop searching for many parts. In these cases, they often simply use another similar

part that is stored in the yard. When this happens, the item that is improperly taken is then missing and must be searched for when requested for delivery.

The same phenomena of compounding errors is apparent in Exhibit 5. This plot shows the average delivery times for all items and misplaced items as a function of misplacement percentage. Again the results of the model increase greater than linearly as the misplacement increases, due the compounding nature of errors in the process.

Exhibit 6 shows the same data as in Exhibit 3, but for the first RFID simulation, which incorporates fixed gate scanners and hand-held scanners in the yard. Exhibit 7 shows the results for the model runs with full RFID implementation: fixed gate scanners, hand-held scanners in the yard, and fixed, tower-based scanners for triangulation.

The results of the model show that improvements can be achieved in each area of time expenditure; inventory, local searching, and remote searching. In addition, the delivery time to the work site and the total amount of lost material can be significantly reduced. The gains that are achievable depend on the level of RFID technology that is implemented and on the original level of misplaced material. The gains achievable by each level of technology insertion will be examined.

Fixed Gate Scanners & Hand-Held Scanners:

Exhibit 8 shows the total improvements provided by the implementation of the first level of RFID technology, as predicted by the simulation model. The incorporation of fixed scanners at the gate and the use of hand-held scanners by yard personnel to identify material reduces the total amount of time dedicated to inventory and search operations. Data is presented for each type of material for both the total 200-hour simulation and averaged for individual items.

The effect on the time spent on inventory is pronounced. Because the process is now fully automated, the driver is no longer required to complete a manual inventory as he enters the yard. After fixed scanners have been placed at the gates, the driver must only slow down, so that the tags on the truck may be scanned. The total time spent on inventory is reduced from 25 hours in the baseline simulation to only 1.5 hours in the first RFID simulation. This equates to a total reduction of 23.5 hours for this simulation, or approximately 0.02 hours per item.

Much more significant, however, are the time reductions for area searches. The times for local searching (searching the actual storage area for an item) is reduced from approximately 210 hours for the baseline case to 40 hours for the first RFID case. This reduction is due to the ability of yard personnel to quickly identify material while searching storage areas. Instead of manually reading individual tags, a worker can now quickly search multiple items until the correct

tag is identified. The result is that the time per item dedicated to searching is reduced from 8.5 minutes to only 1.6 minutes.

Similarly, the time dedicated to remote searching can be similarly reduced. Yard workers with hand-held scanners can more quickly scan areas to determine that a desired item is not present and move on with the search. The total time dedicated to remote searching is, of course, heavily dependent on the level of material that is misplaced. The time expenditure is reduced from 38.2 hours to 15.2 hours for a 5% level of lost material and from 270.0 hours to 109.5 hours for a 30% level of lost material.

The total manhour reductions for this simulation, provided by the fixed gate scanners and hand-held scanners, is from 217 hours to 351 hours for the 5% and 30% cases, respectively. This equates to a reduction of between 0.44 and 0.58 hours for every item passing through the shipyard.

Fixed Gate Scanners, Hand-Held Scanners, and Fixed Towers:

Exhibit 9 shows the total improvements obtainable by full implementation of the proposed RFID technology over the baseline case. In this case, along with the changes made for the first RFID simulation, fixed, tower-based scanners have been added to allow for triangulation of tag locations. In this simulation, yard personnel can instantly determine the location of any piece of material by querying its RFID tag from the fixed scanners. Locations are provided on

portable displays and the workers can proceed directly to the correct storage area. No material is considered to be lost, in this case, since all items can be quickly found, regardless of whether they are stored in the proper location.

No additional time savings are provided in the times dedicated to inventory or local searching by the fixed RFID towers. These functions must still be performed using the fixed gate and hand-held scanners.

Additional time savings are provided, however, in the area of remote search times. By quickly locating lost material, remote searching is eliminated. Total manhours for remote searching for the 5% case is reduced from 38.2 manhours for the baseline case and 15.2 hours for the first RFID case, to 0 hours for full implementation. For the 30% case, the manhours are reduced from 270.0 hours for the baseline case and 109.5 hours for the first RFID case to 0 hours for the full implementation.

The total manhours eliminated by the full implementation of RFID technology is between 232.7 hours for the 5% case and 460.4 hours for the 30% case, over the baseline. This represents an additional reduction over the first RFID case of between 15.0 and 109.0 manhours, respectively.

Financial Calculations:

Exhibits 10 and 11 show the calculation of the financial benefits that can be expected from the implementation of RFID technology. Exhibit 10 converts the total manhours saved in the material control simulation for each level of RFID implementation to dollars saved per year in yard operation. Calculations are presented for savings at each increment of material misplacement. Cost calculations are performed assuming a 2000 hour man-year and an average yearly cost to the shipyard for a worker of $70,000. It should be noted that these cost reductions calculations include only savings attributable to direct reductions in manhours dedicated to inventory and searching. Additional savings will be present due to reduced delays on the waterfront and reduction in lost material. These additional savings will be discussed later in this chapter.

The yearly cost savings resulting from the implementation of fixed gate scanners and hand-held scanners ranges from $76,008 to $122,985, depending in the material misplacement rate. The values for the full RFID implementation range from $81,458 to $161,124 over the baseline. This equates to an additional savings of between $5,450 and $38,139, attributable to the addition of the tower-based scanners.

Exhibit 11 shows the calculations for the net present value (NPV) of the savings from RFID implementation. The NPV is calculated based on an inflation rate of 4%, an internal rate of return for the company of 12%, over a period of 10 years.

The NPV value is significant to this analysis because it represents the maximum amount that should be spent on system implementation in order to produce a net financial benefit to the yard.

The total annual saving due to implementation of the first level of RFID technology is between $497,283 and $804,631, for the 5% and 30% cases, respectively. The savings over the baseline for the full implementations are $532,940 for the 5% case and $1,054,156 for the 30% case. This equates to an additional total savings of $35,651 and $249,525 for the tower-based scanners.

Material Misplacement Rate:

Results from the simulation exercise have been presented over a fairly wide range of possible material misplacement rates. This was done because the actual level of material misplacement in the yard is difficult to measure and can vary substantially based on current conditions. However, in order to determine what the actual economic benefits of RFID implementation are, an estimate should be made as to what material misplacement rate the yard is likely to experience in actual operation, over a long period of time.

As mentioned in Chapter 8, the total material misplacement rate is a function of two separate rates; the rate of material misplaced due to full storage locations and the rate of material misplaced due to driver error. In the simulation, the rate

of driver error was controlled as an input, while the rate of full locations was determined by the software as a result of the simulation.

In running the simulation models, the 5% total material misplacement could be achieved only by setting the driver error rate to 0%. It can be reasonably assumed that in actual yard operation the driver error rate will be at least 5% under normal conditions. This means that in only 1 out of 20 deliveries the driver would make an error in either inventory, material identification, or material delivery. A 5% driver error rate results in an overall misplacement rate of 10%. This is the value that should be used for determining the minimum financial benefits to the yard.

It is likely, however, that the total average misplacement rate will be somewhat higher than this though. During times of heavy material flow, low worker availability, or poor weather, the driver error rate is likely to climb. A realistic estimate of the average material misplacement rate, over the long-term, is 15% - 20%.

Material Delivery Time:

There are other improvements in material control operations, aside from reduction in manhours, that result from the implementation of RFID technology. Another important factor is the reduction in the time that it takes for material to be delivered to the worksite. Not only does increased delivery time mean that more

time is being spent by yard personnel in the search for missing items, but more importantly, it means that waterfront personnel are being kept waiting and work on the ship is being delayed. As yard personnel search for lost material, yard workers are often unable to proceed with construction. The result can be that many workers are kept idle, future construction schedules can be disrupted, and final delivery can be delayed. These are all very expensive prospects to the yard.

While an actual financial analysis of material delays was not performed as part of this report, by examining the results from the simulation model, some obvious improvements in this area can be observed. In the baseline case the average delivery time for all material ranged from 16.50 minutes for the 5% misplacement case to 40.34 for the 30% case. The numbers were much worse for material that had been misplaced. The average delivery time for lost items was 40.6 minutes for the 5% case and 64.0 minutes for the 30%.

In many cases the delivery times for some items are much longer than this. Exhibit 12 shows the distribution of delivery times for misplaced items in the 15% misplacement case. It can be seen that delivery times range up to 220 minutes and that out of the 150 misplaced, 25, or 16%, took over 100 minutes to be delivered.

The implementation of gate scanners and hand-held scanners alone has a major impact on delivery time. Because local searches can now be conducted faster, and storage area volumes are lower, delivery times are improved. The average delivery time for all items is reduced to 11.2 minutes for the 5% case and 26.9 minutes for the 30% case, a reduction of 5.3 and 13.4 minutes respectively. For misplaced items the delivery time is reduced to 15.95 minutes for the 5% case and 42.5 minutes for the 30% case. This equates to a reduction of between 21.5 and 24.6 minutes over the baseline.

The fixed tower scanners provide further improvement in delivery time. By eliminating lost material the delivery time is reduced to 9.0 minutes for all items. This is a reduction of between 7.5 and 31.3 minutes over the baseline case and between 2.2 and 27.9 minutes over the first RFID case.

Lost Material:

An additional area in which RFID can contribute to the reduction of shipyard costs is in decreasing the amount of material that is ultimately lost in the yard. Inevitably, when items are misplaced in the yard, some material will never be recovered. As search times become long, yard personnel are likely to either use a replacement part that is accessible or will reorder the item for future delivery. In both cases, the shipyard must pay for additional material and will experience further delays due to unavailable material.

The likelihood of an item becoming permanently lost is related to the amount of time that must be spent in the search process. As search times extend beyond an hour or two hours, it is likely that workers will abandon the search and use a substitute item.

Exhibit 13 shows the average number of items that require searches longer than one, two, and three hours for each increment of material misplacement. At a 15% material misplacement rate, 74 out of the 225 lost items will take longer than 1 hour to recover, 26 items will take longer than 2 hours, and 5 will take longer than three hours.

Although the exact number of items that will be permanently lost is difficult to determine, it is obvious from this data that many of these long-period searches would be abandoned. With the high value of material that passes through the yard, the costs could be significant. With the full implementation of RFID, searches take place much more quickly and lost material can virtually be eliminated.

The total cost savings attributable to both reduction in waiting by waterfront personnel and elimination of lost material is difficult to determine. However, it is likely to be at least as great as the savings directly attributable to reduced manhours in the material control process. These additional potential savings should be considered when determining the overall economic benefits of RFID.

Chapter 10: Other Possible RFID Shipyard Implementation

There are several other options for using RFID to improve shipyard material control processes that were not included in the initial BIW plan or in the process simulation. These options were not included either because the type of tag used at BIW would not support the operation or because the implementation would require too great a modification to the current system. These functions, however, do still hold promise for future use.

In a system where RFID tags have the ability to store actual data, rather than just an ID number, the system could be used improve operation of the JIT material delivery system and the efficiency of material handling in the yard. The tags themselves could be programmed with the required delivery time and location of associated items. The system could then be configured so that tags could actually call yard workers for pick-up when items needed to be delivered in the yard. This would eliminate the need for controllers to constantly monitor delivery times and dispatch workers to collect material.

Data could also be contained on tags concerning the handling, assembly and testing of material and equipment. Currently, data regarding these subjects is stored in multiple databases in the yard. In order to examine testing data, yard personnel must find the ID number of a piece of equipment, must cross reference the testing data, and then must recall that data from storage. Errors can be

made in the recording and recovery of data that lead to mistakes in testing or in improper installation. By recording data directly on tags, it would be instantly available to yard personnel when needed, without problems in misidentification of material.

Handling instructions could also be coded directly onto RFID tags. Special instructions regarding required moving procedures or environmental conditions, which might be overlooked, can be programmed on a tag. When queried, the tag would then give personnel all the information required for proper handling and storage procedures.

Similarly, it could be possible to increase construction efficiency in a shipyard if other pertinent construction data was stored directly on RFID tags. Assembly unit identification, related drawing numbers, or welding requirements could be stored on tags. This data could then be accessed by workers without having to access central databases or drawings. This could save man-hours and wasted time in the construction process.

It is also possible that some condition monitoring via RFID could be used in the shipyard on a small scale to improve material handling. Certain, sensitive pieces of machinery and equipment require strict environmental controls for storage. Monitoring devices, connected to RFID could check conditions to ensure that requirements are being met. If conditions, such as temperature or humidity,

exceeded allowable ranges, the tag could be programmed to transmit a warning to the central database.

While all of these functionalities are possible using existing RFID technology, it would be necessary to analyze the potential gains to the shipyard before developing an implementation plan. As with the proposed system at BIW, the operations of the modified processes would have to be compared with the existing processes to determine savings in man-hours and material.

Chapter 11: Synergy with Shipboard Activities

The use of RFID technology in the shipyard could have other significant benefits to the U.S. Navy, aside from just reduction in acquisition cost. There is a potential for synergy in tag use between activities in the shipyard and activities performed during the life cycle of the ship that could further reduce the overall cost of ownership.

RFID has the potential to dramatically improve configuration control on-board Navy ships. If all equipment was tagged at the shipyard during construction, with tags that could be read by both shipyard and Navy personnel, the system could be used to identify material as it was brought on and taken off the ship, creating a running inventory of equipment. More importantly, the tags could be queried to create an instant inventory of the present equipment in a particular space. Often, when repair or refit work is performed, extensive surveys of equipment and systems must first take place to identify the current condition of the ship. By using RFID tags this process could be greatly streamlined.

Important data concerning equipment performance and maintenance could also be stored on RFID tags attached directly to the machinery. This could begin with results from shipyard testing and could include maintenance instructions and schedules, spare part identification, and repair history. Presently, this type of data is stored in a variety of formats and locations, which are often difficult to find

and access, if it is stored at all. If this information was stored locally and accurately, the crew would have a much easier job accessing required data and would be more likely to perform maintenance and repairs in a correct and timely fashion.

RFID could also be beneficial on-board ship in improving the control of spares and stores. Again, if the shipyard was to tag and scan items such as spares and tools at the initial fit out, and the Navy were to continue to use the same system during the replenishment and use of these items, then a running inventory of material could be maintained. Spot inventories could be performed, using portable scanners in storage spaces and the tags could be used to quickly search for material on-board ship. It would also be possible to connect the inventory database to the ordering system to automate material ordering.

The potential for use of RFID on-board ship is extensive. With the vast quantity of equipment and the large numbers of tasks that must be performed, there is the need for automated identification. It is important, however, that shipboard RFID activities be coordinated with those that take-place in the shipyard. The greatest potential gains occur when the technology is used throughout the life cycle of the ship. The Navy must ensure that similar technology and processes are used by all the parties involved in conjunction with a coordinated plan for shipyard and shipboard use.

Chapter 12: Conclusions

The simulation model has shown that the use of RFID technology can provide improvements to shipyard operations and reductions in construction cost. Time spent on material control in the processes of inventory, item recovery, and searching for lost items can all be substantially reduced. In addition, delays in construction due to delinquent material arrival can be reduced and the loss of high-value material can be eliminated.

The financial analysis presented in Chapter 9 provides some indication of the potential benefits that could be provided to shipyards by the incorporation of RFID technology. If we assume an average material misplacement rate of 15%, the total NPV of the cost reductions associated with the implementation of gate scanners and hand held scanners is $604,888. This value assumes a period of 10 years and a discount rate of 12%. The NPV of cost reductions for the full implementation of RFID is $716,091, an additional savings of $112,073. Additional cost reductions will be derived from the RFID implementation with the reduction of delivery times and the elimination of lost material.

The implementation costs for the first level of RFID implementation, gate scanners and hand-held scanners is likely to be fairly reasonable. The scanners themselves are not the major expenditure, but rather the acquisition and installation of the tags themselves is likely to drive the total cost. Each pallet and storage box used in the yard must have a tag permanently affixed. In addition, a

large pool of tags, to be used to identify large, individually stored pieces of equipment, must also be provided. However, since no major infrastructure changes are required, the total cost could be quite reasonable.

The addition of the fixed tower scanners, however, could be a significant cost. Depending of the achievable range of the tags, several towers could be required to cover all possible storage areas. While the cost for the scanners themselves would not be great, the cost of constructing the towers, or finding alternate scanner locations, could be large.

Given the relative benefits and costs of each level of RFID implementation, it is recommended that BIW proceed with the financial analysis of the first level of RFID implementation. The majority of the time reductions can be obtained with the simpler installation, and the total acquisition costs should be much less. A plan for implementation should be developed and the total acquisition costs determined.

If the total implementation cost of the gate scanners and hand-held scanners is found to be less than the $605,000 cost savings NVP, then BIW should proceed with the installation of the equipment. If the implementation cost is somewhat greater than the $605,000 direct savings, then additional analysis should be conducted to examine the reduced costs due to reduced waterfront delays and

lost material. These factors could make the overall project beneficial, even if the direct manhour reductions do not justify the capital outlay.

If the project proceeds, careful monitoring of manhour reductions and material misplacements, using the new system, should be conducted to verify the benefits. If the technology is found to be effective, then further analysis of the fixed, towed-based RFID scanners should then be conducted.

In addition, if the first implementation of RFID technology is found to be effective, further research should then take place concerning other applications of RFID technology, both in the shipyard and in conjunction with on-board activities.

Other large shipyards, both naval and commercial, could achieve benefits similar to those predicted for BIW, with the implementation of automatic identification technology. The analysis in this report shows that complex material control procedures can by greatly improved using RFID or similar technology. These improvements should be pursued by all major shipyards in an effort to modernize procedures and reduce acquisition costs.

References:

Dunlap, Gary, "Applied Information Technology For Ship Design, Production and Lifecycle Support: A Total Systems Approach", Master's Thesis, MIT, 1999.

Bateman, Bowden, Gogg, Harrell, and Mott, System Improvement Using Simulation, 5th Ed., ProModel Corporation, Orem, Utah, 1997.

ProModel User's Guide, ProModel Corporation, Orem, Utah, 1999.

ProModel Reference Guide, ProModel Corporation, Orem, Utah, 1999.

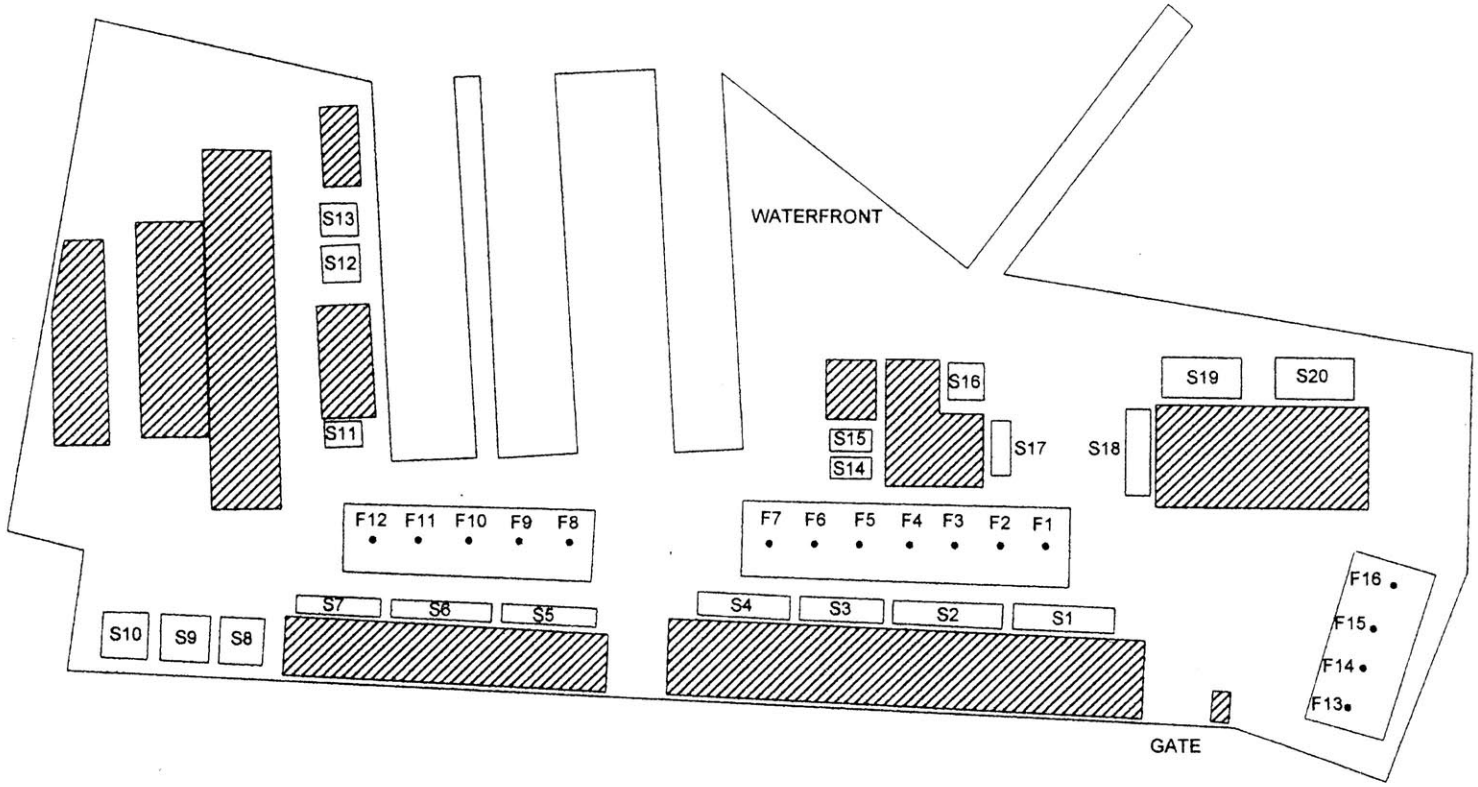
BIW Drawing No. 7354, "BIW Site Map", pp. 211, 07-01-98.

BIW Drawing No. 9200, "East Brunswick Manufacturing Facility Site Map", 05-28-94.

BIW Drawing No. 2064, "Harding Plant Site Map", pp. 202, 06-11-83.

Exhibits

Exhibit 1: Model Layout



F = FLAT-BED STORAGE LOCATIONS
S = GROUND STORAGE LOCATIONS

Exhibit 2: Variables and Input Parameters

Entries into System:

Ground Storage Items:

1000 total entries into system
Units per truckload = (8,1.5) = 8 units per truck with a standard deviation of 1.5 units
122 total truckloads
Period between trucks = (80,15) = 80 minutes with a standard deviation of 15 minutes

Flat-Bed Storage Items:

500 total entries into system
Units per flat-bed = (10,2) = 10 units per flat-bed with a standard deviation of 2 units
50 total flat-bed trailers
Period between trucks = (100,20) = 100 minutes with a standard deviation of 20 minutes

Time to Perform Inventory at Gate:

Baseline:

Inventory Time (min) = (2.0, 0.5) + N * (1.0, 0.25)

Fixed Time =2.0 minutes with a standard deviation of 0.5 minutes.
Variable Time = 1.0 minute per item on truck with a standard deviation of 0.25 minutes
N = Number of items on truck

RFID:

Search Time (min) = (0.50, 0.10)

Fixed Time = 0.5 minutes with a standard deviation of 0.1 minutes

Time to Search a Storage Area:

Baseline:

Search Time (min) = (3.0, 1.0) + N * (1.0, 0.5)

Fixed Time =3.0 minutes with a standard deviation of 1.0 minute.
Variable Time = 1.0 minute per item in location with a standard deviation of 0.5 minutes
N = Number of items currently in storage location

RFID:

Search Time (min) = (0.50, 0.10) + N * (0.20, 0.05)

Fixed Time = 0.5 minutes with a standard deviation of 0.1 minutes
Variable Time = 0.2 minute per item in location with a standard deviation of 0.05 minutes
N = Number of items currently in storage location

Exhibit 3: Model Results
Baseline

Ground Storage Units:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Total Time - Inventory	min	995.75	1002.99	998.43	1006.28	996.29	994.70
Total Time - Local Searching	min	8694.64	8559.52	8581.77	8552.75	8612.62	8611.51
Total Time - Remote Searching	min	2157.79	4322.79	6168.14	8136.26	10661.13	12688.73
Time per Unit - Inventory	min	1.00	1.00	1.00	1.01	1.00	0.99
Time per Unit - Local Search	min	8.69	8.56	8.58	8.55	8.61	8.61
Time per Unit - Remote Search	min	2.16	4.32	6.17	8.04	10.66	12.69
Time per Misplaced Unit - Remote Search	min	43.16	43.23	41.12	40.68	42.64	42.30
Units Misplaced		50	100	150	200	250	300
Avg. Time to Delivery - All	min	18.84	23.65	29.14	35.85	43.19	50.04
Avg. Time to Delivery - Misplaced Units	min	55.03	57.64	62.14	68.08	72.77	77.34

Flat-Bed Storage Units:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Total Time - Inventory	min	499.01	496.00	501.12	493.61	498.67	498.02
Total Time - Local Searching	min	4013.46	3905.35	3892.27	3972.34	3918.11	3832.51
Total Time - Remote Searching	min	133.91	464.07	1174.70	1873.19	2482.36	3526.11
Time per Unit - Inventory	min	1.00	0.99	1.00	0.99	1.00	1.00
Time per Unit - Local Search	min	8.03	7.81	7.78	7.94	7.84	7.67
Time per Unit - Remote Search	min	0.27	0.93	2.35	3.75	4.96	7.05
Time per Misplaced Unit - Remote Search	min	5.36	9.28	15.66	18.73	19.86	23.51
Units Misplaced		25	50	75	100	125	150
Avg. Time to Delivery - All	min	11.83	12.34	14.79	15.54	16.04	20.92
Avg. Time to Delivery - Misplaced Units	min	11.67	16.59	23.74	26.17	28.15	37.26

Total Units:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Total Time - Inventory	min	1494.77	1498.99	1499.55	1499.90	1494.95	1492.71
Total Time - Local Searching	min	12708.10	12464.87	12474.04	12525.09	12530.73	12444.02
Total Time - Remote Searching	min	2291.70	4786.86	7342.84	10009.45	13143.49	16214.84
Time per Unit - Inventory	min	1.00	1.00	1.00	1.00	1.00	1.00
Time per Unit - Local Search	min	8.47	8.31	8.32	8.35	8.35	8.30
Time per Unit - Remote Search	min	1.53	3.19	4.90	6.67	8.76	10.81
Time per Misplaced Unit - Remote Search	min	30.56	31.91	32.63	33.36	35.05	36.03
Units Misplaced		75	150	225	300	375	450
Avg. Time to Delivery - All	min	16.50	19.88	24.36	29.08	34.14	40.34
Avg. Time to Delivery - Misplaced Units	min	40.57	43.95	49.34	54.11	57.90	63.98

Note: Values for average inventory times and local search times are essentially steady at all levels of material misplacement. Variations occur in individual data sets due to the standard deviations in process times used in the simulation.

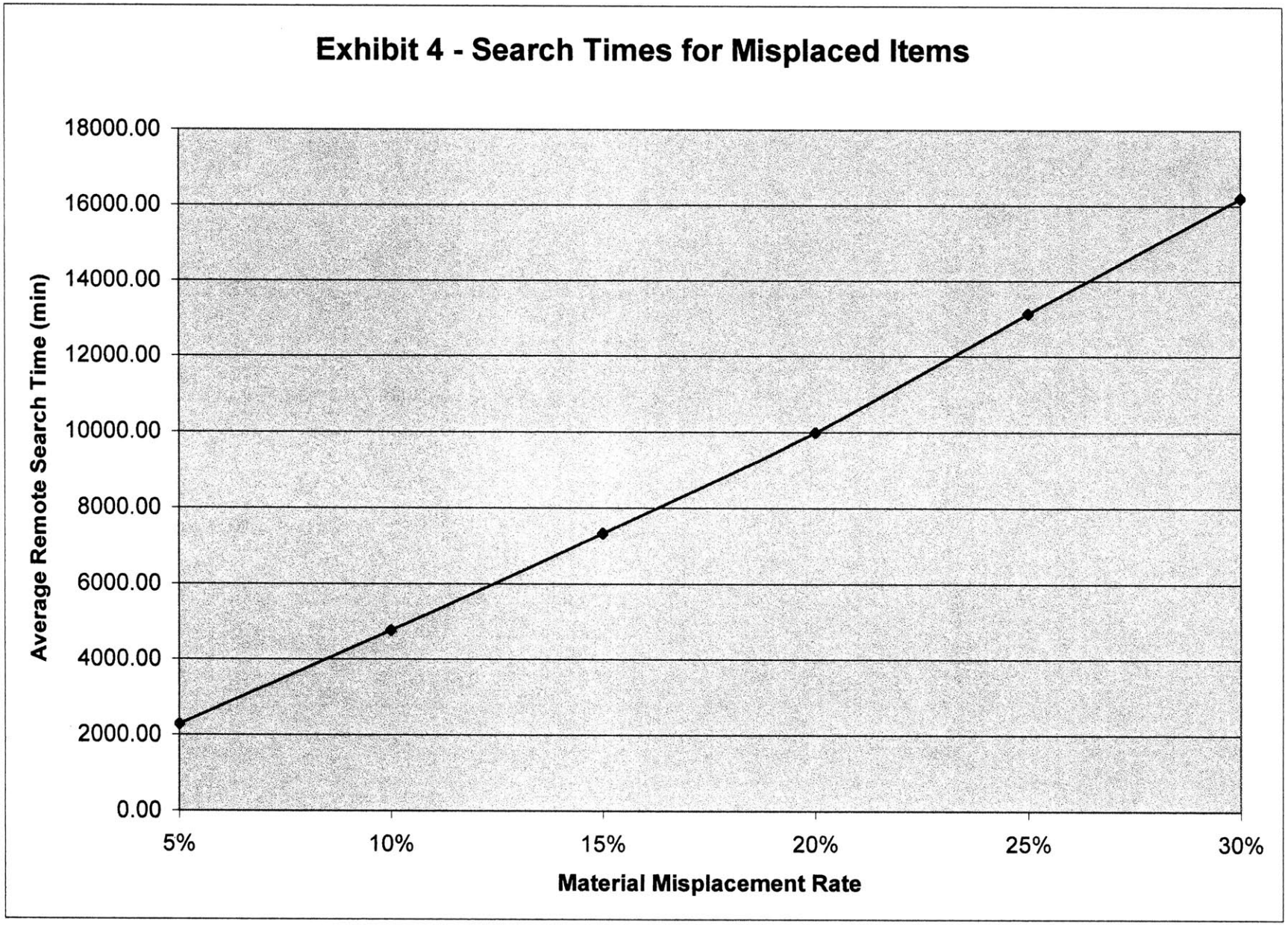


Exhibit 4 - Search Times for Misplaced Items
18000.00
16000.00
14000.00
12000.00
10000.00
8000.00
6000.00
4000.00
2000.00
0.00
Average Remote Search Time (min)
5%
10%
15%
20%
25%
30%
Material Misplacement Rate

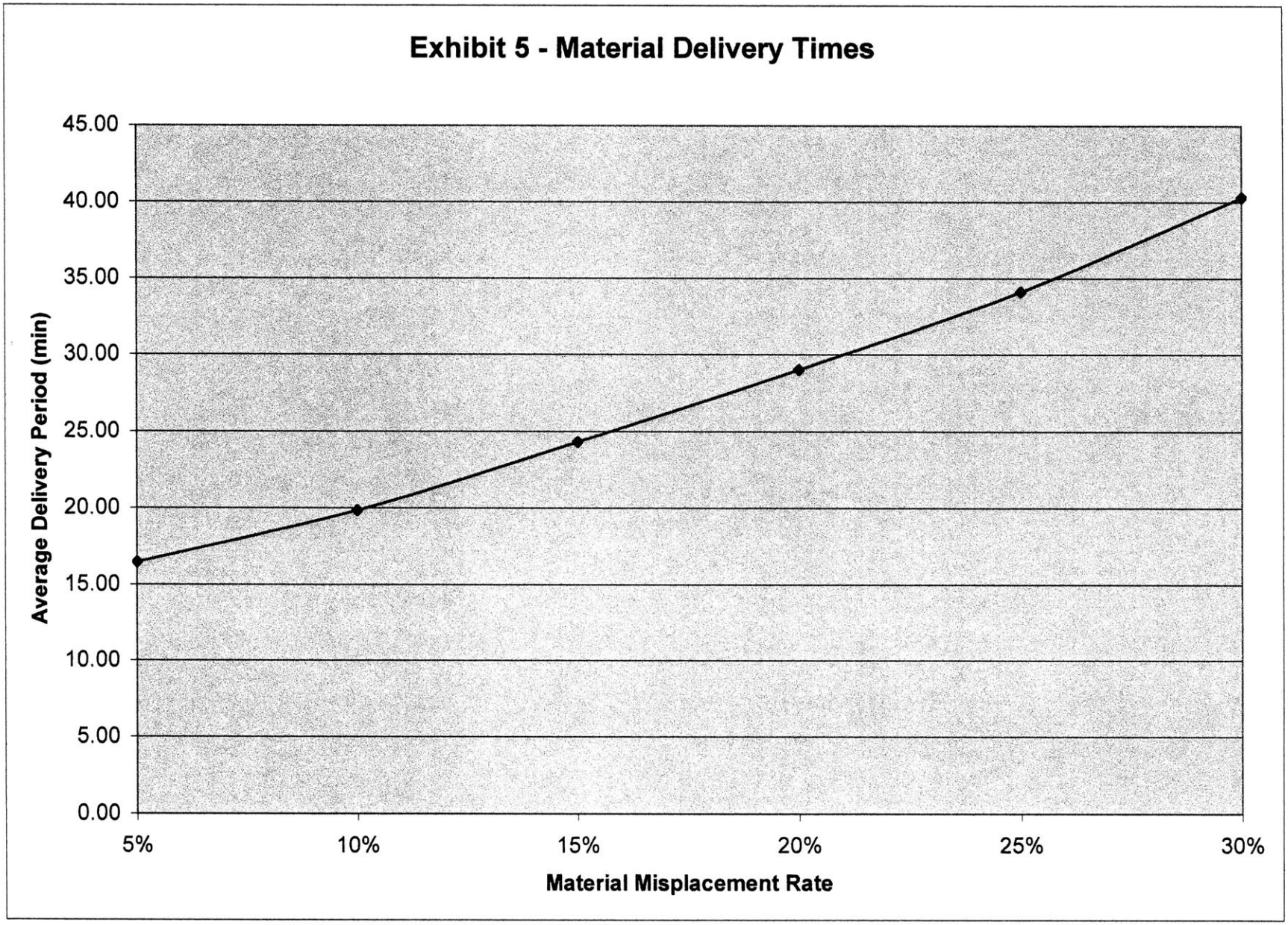


Exhibit 5 - Material Delivery Times
45.00
40.00
35.00
30.00
25.00
20.00
15.00
10.00
5.00
0.00
Average Delivery Period (min)
5%
10%
15%
20%
25%
30%
Material Misplacement Rate

Exhibit 6: Model Results
Fixed Gate Scanners + Hand-held Scanners

Ground Storage Units:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Total Time - Inventory	min	61.74	62.19	61.90	62.39	61.77	61.67
Total Time - Local Searching	min	1678.27	1652.19	1658.41	1650.88	1662.44	1662.22
Total Time - Remote Searching	min	863.12	1685.89	2498.10	3190.40	4243.13	5100.87
Time per Unit - Inventory	min	0.06	0.06	0.06	0.06	0.06	0.06
Time per Unit - Local Search	min	1.68	1.65	1.66	1.65	1.66	1.66
Time per Unit - Remote Search	min	0.86	1.69	2.50	3.19	4.24	5.10
Time per Misplaced Unit - Remote Search	min	17.26	16.86	16.65	15.95	16.97	17.00
Units Misplaced		50	100	150	200	250	300
Avg. Time to Delivery - All	min	12.20	15.31	18.86	23.21	27.96	32.40
Avg. Time to Delivery - Misplaced Units	min	20.85	24.84	30.29	37.59	44.22	51.00

Flat-Bed Storage Units:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Total Time - Inventory	min	29.94	29.76	30.07	29.62	29.92	29.88
Total Time - Local Searching	min	782.62	761.54	758.99	774.61	764.03	747.34
Total Time - Remote Searching	min	49.00	175.68	459.61	748.47	996.09	1466.50
Time per Unit - Inventory	min	0.06	0.06	0.06	0.06	0.06	0.06
Time per Unit - Local Search	min	1.57	1.52	1.52	1.55	1.53	1.49
Time per Unit - Remote Search	min	0.10	0.35	0.92	1.50	1.99	2.93
Time per Misplaced Unit - Remote Search	min	1.96	3.51	6.13	7.48	7.97	9.78
Units Misplaced		25	50	75	100	125	150
Avg. Time to Delivery - All	min	9.05	9.44	11.32	11.89	12.27	16.00
Avg. Time to Delivery - Misplaced Units	min	6.15	9.05	14.26	15.95	17.99	25.45

Total Units:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Total Time - Inventory	min	91.68	91.95	91.97	92.01	91.69	91.55
Total Time - Local Searching	min	2460.89	2413.73	2417.40	2425.49	2426.47	2409.56
Total Time - Remote Searching	min	912.11	1861.57	2957.71	3938.87	5239.22	6567.36
Time per Unit - Inventory	min	0.06	0.06	0.06	0.06	0.06	0.06
Time per Unit - Local Search	min	1.64	1.61	1.61	1.62	1.62	1.61
Time per Unit - Remote Search	min	0.61	1.24	1.97	2.63	3.49	4.38
Time per Misplaced Unit - Remote Search	min	12.16	12.41	13.15	13.13	13.97	14.59
Units Misplaced		75	150	225	300	375	450
Avg. Time to Delivery - All	min	11.15	13.35	16.35	19.43	22.73	26.93
Avg. Time to Delivery - Misplaced Units	min	15.95	19.58	24.95	30.38	35.48	42.48

Note: Values for average inventory times and local search times are essentially steady at all levels of material misplacement. Variations occur in individual data sets due to the standard deviations in process times used in the simulation.

Exhibit 7: Model Results
Fixed Gate Scanners + Hand-held Scanners + Fixed Towers

Ground Storage Units:

Total Time - Inventory	min	61.57
Total Time - Local Searching	min	1660.99
Total Time - Remote Searching	min	0.00

Time per Unit - Inventory	min	0.06
Time per Unit - Local Search	min	1.66
Time per Unit - Remote Search	min	0.00
Time per Misplaced Unit - Remote Search	min	n/a

Units Misplaced		0
Avg. Time to Delivery - All	min	9.71
Avg. Time to Delivery - Misplaced Units	min	n/a

Flat-Bed Storage Units:

Total Time - Inventory	min	29.54
Total Time - Local Searching	min	783.28
Total Time - Remote Searching	min	0.00

Time per Unit - Inventory	min	0.06
Time per Unit - Local Search	min	1.57
Time per Unit - Remote Search	min	0.00
Time per Misplaced Unit - Remote Search	min	n/a

Units Misplaced		0
Avg. Time to Delivery - All	min	7.65
Avg. Time to Delivery - Misplaced Units	min	n/a

Total Units:

Total Time - Inventory	min	86.13
Total Time - Local Searching	min	2444.27
Total Time - Remote Searching	min	0.00

Time per Unit - Inventory	min	0.06
Time per Unit - Local Search	min	1.63
Time per Unit - Remote Search	min	0.00
Time per Misplaced Unit - Remote Search	min	n/a

Units Misplaced		0
Avg. Time to Delivery - All	min	9.02
Avg. Time to Delivery - Misplaced Units	min	n/a

Exhibit 8: Manhour Reductions
Baseline to Fixed Gate Scanners + Hand-held Scanners

Ground Storage Units:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Total Time - Inventory	hr	15.57	15.68	15.61	15.73	15.58	15.55
Total Time - Local Searching	hr	116.94	115.12	115.39	115.03	115.84	115.82
Total Time - Remote Searching	hr	21.58	43.95	61.17	82.43	106.97	126.46
Total	hr	154.08	174.75	192.17	213.19	238.38	257.84
Time per Unit - Inventory	hr	0.02	0.02	0.02	0.02	0.02	0.02
Time per Unit - Local Search	hr	0.12	0.12	0.12	0.12	0.12	0.12
Time per Unit - Remote Search	hr	0.02	0.04	0.06	0.08	0.11	0.13
Time per Misplaced Unit - Remote Search	hr	0.43	0.44	0.41	0.41	0.43	0.42
Total	hr	0.57	0.60	0.58	0.61	0.65	0.66

Flat-Bed Storage Units:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Total Time - Inventory	hr	7.82	7.77	7.85	7.73	7.81	7.80
Total Time - Local Searching	hr	53.85	52.40	52.22	53.30	52.57	51.42
Total Time - Remote Searching	hr	1.42	4.81	11.92	18.75	24.77	34.33
Total	hr	63.08	64.97	71.99	79.77	85.15	93.55
Time per Unit - Inventory	hr	0.02	0.02	0.02	0.02	0.02	0.02
Time per Unit - Local Search	hr	0.11	0.10	0.10	0.11	0.11	0.10
Time per Unit - Remote Search	hr	0.00	0.01	0.02	0.04	0.05	0.07
Time per Misplaced Unit - Remote Search	hr	0.06	0.10	0.16	0.19	0.20	0.23
Total	hr	0.17	0.21	0.29	0.33	0.35	0.40

Total Units:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Total Time - Inventory	hr	23.38	23.45	23.46	23.46	23.39	23.35
Total Time - Local Searching	hr	170.79	167.52	167.61	168.33	168.40	167.24
Total Time - Remote Searching	hr	22.99	48.75	73.09	101.18	131.74	160.79
Total	hr	217.16	239.72	264.16	292.97	323.53	351.38
Time per Unit - Inventory	hr	0.02	0.02	0.02	0.02	0.02	0.02
Time per Unit - Local Search	hr	0.11	0.11	0.11	0.11	0.11	0.11
Time per Unit - Remote Search	hr	0.02	0.03	0.05	0.07	0.09	0.11
Time per Misplaced Unit - Remote Search	hr	0.31	0.33	0.32	0.34	0.35	0.36
Total	hr	0.44	0.47	0.49	0.52	0.55	0.58

Note: Values for average inventory times and local search times are essentially steady at all levels of material misplacement. Variations occur in individual data sets due to the standard deviations in process times used in the simulation.

Exhibit 9: Manhour Reductions
Baseline to Fixed Gate Scanners + Hand-held Scanners + Fixed Towers

Ground Storage Units:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Total Time - Inventory	hr	15.57	15.69	15.61	15.75	15.58	15.55
Total Time - Local Searching	hr	117.23	114.98	115.35	114.86	115.86	115.84
Total Time - Remote Searching	hr	35.96	72.05	102.80	135.60	177.69	211.48
Total	hr	168.76	202.71	233.76	266.21	309.12	342.87
Time per Unit - Inventory	hr	0.02	0.02	0.02	0.02	0.02	0.02
Time per Unit - Local Search	hr	0.12	0.11	0.12	0.11	0.12	0.12
Time per Unit - Remote Search	hr	0.04	0.07	0.10	0.13	0.18	0.21
Total	hr	0.17	0.20	0.23	0.26	0.31	0.34

Flat-Bed Storage Units:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Total Time - Inventory	hr	7.82	7.77	7.86	7.73	7.82	7.81
Total Time - Local Searching	hr	53.84	52.03	51.82	53.15	52.25	50.82
Total Time - Remote Searching	hr	2.23	7.73	19.58	31.22	41.37	58.77
Total	hr	63.89	67.54	79.25	92.11	101.44	117.40
Time per Unit - Inventory	hr	0.02	0.02	0.02	0.02	0.02	0.02
Time per Unit - Local Search	hr	0.11	0.10	0.10	0.11	0.10	0.10
Time per Unit - Remote Search	hr	0.00	0.02	0.04	0.06	0.08	0.12
Total	hr	0.13	0.14	0.16	0.18	0.20	0.23

Total Units:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Total Time - Inventory	hr	23.48	23.55	23.56	23.56	23.48	23.44
Total Time - Local Searching	hr	171.06	167.01	167.16	168.01	168.11	166.66
Total Time - Remote Searching	hr	38.20	79.78	122.38	166.82	219.06	270.25
Total	hr	232.74	270.34	313.10	358.40	410.65	460.35
Time per Unit - Inventory	hr	0.02	0.02	0.02	0.02	0.02	0.02
Time per Unit - Local Search	hr	0.11	0.11	0.11	0.11	0.11	0.11
Time per Unit - Remote Search	hr	0.03	0.05	0.08	0.11	0.15	0.18
Total	hr	0.16	0.18	0.21	0.24	0.27	0.31

Note: Values for average inventory times and local search times are essentially steady at all levels of material misplacement. Variations occur in individual data sets due to the standard deviations in process times used in the simulation.

Exhibit 10: Cost Reductions

Manhours Reduced for Simulation Period:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Fixed Gate Scanners + Hand-held Scanners	hr	217.16	239.72	264.16	292.97	323.53	351.38
Fixed Gate Scanners + Hand-held Scanners + Fixed Towers	hr	232.74	270.34	313.10	358.40	410.65	460.35

Manhours Reduced per Year:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Fixed Gate Scanners + Hand-held Scanners	hr	2171.65	2397.25	2641.56	2929.68	3235.30	3513.85
Fixed Gate Scanners + Hand-held Scanners + Fixed Towers	hr	2327.36	2703.39	3131.00	3584.01	4106.46	4603.53

Man-Years Reduced per Year:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Fixed Gate Scanners + Hand-held Scanners	man-y	1.09	1.20	1.32	1.46	1.62	1.76
Fixed Gate Scanners + Hand-held Scanners + Fixed Towers	man-y	1.16	1.35	1.57	1.79	2.05	2.30

Cost Reduction per Year:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Fixed Gate Scanners + Hand-held Scanners	$	$76,008	$83,904	$92,455	$102,539	$113,236	$122,985
Fixed Gate Scanners + Hand-held Scanners + Fixed Towers	$	$81,458	$94,618	$109,585	$125,440	$143,726	$161,124

* - Assumes $70,000/man-year cost to shipyard

Notes: Calculations do not include cost reductions due to elimination of delays in material arrival at work area
Calculations do not include cost reductions due to elimination of lost material

Exhibit 11: Net Present Values of Cost Reductions

Current Cost Reduction per Year:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Fixed Gate Scanners + Hand-held Scanners	$	$76,008	$83,904	$92,455	$102,539	$113,236	$122,985
Fixed Gate Scanners + Hand-held Scanners + Fixed Towers	$	$81,458	$94,618	$109,585	$125,440	$143,726	$161,124

NPV of Cost Reductions:

		Percentage Misplaced Items					
		5%	10%	15%	20%	25%	30%
Fixed Gate Scanners + Hand-held Scanners	$	$497,283	$548,943	$604,888	$670,863	$740,848	$804,631
Fixed Gate Scanners + Hand-held Scanners + Fixed Towers	$	$532,940	$619,039	$716,961	$820,693	$940,329	$1,054,156

Inflation = 4%
Internal Rate of Return = 12%
Period of 10 Years

Notes: Calculations do not include cost reductions due to elimination of delays in material arrival at work area
Calculations do not include cost reductions due to elimination of lost material

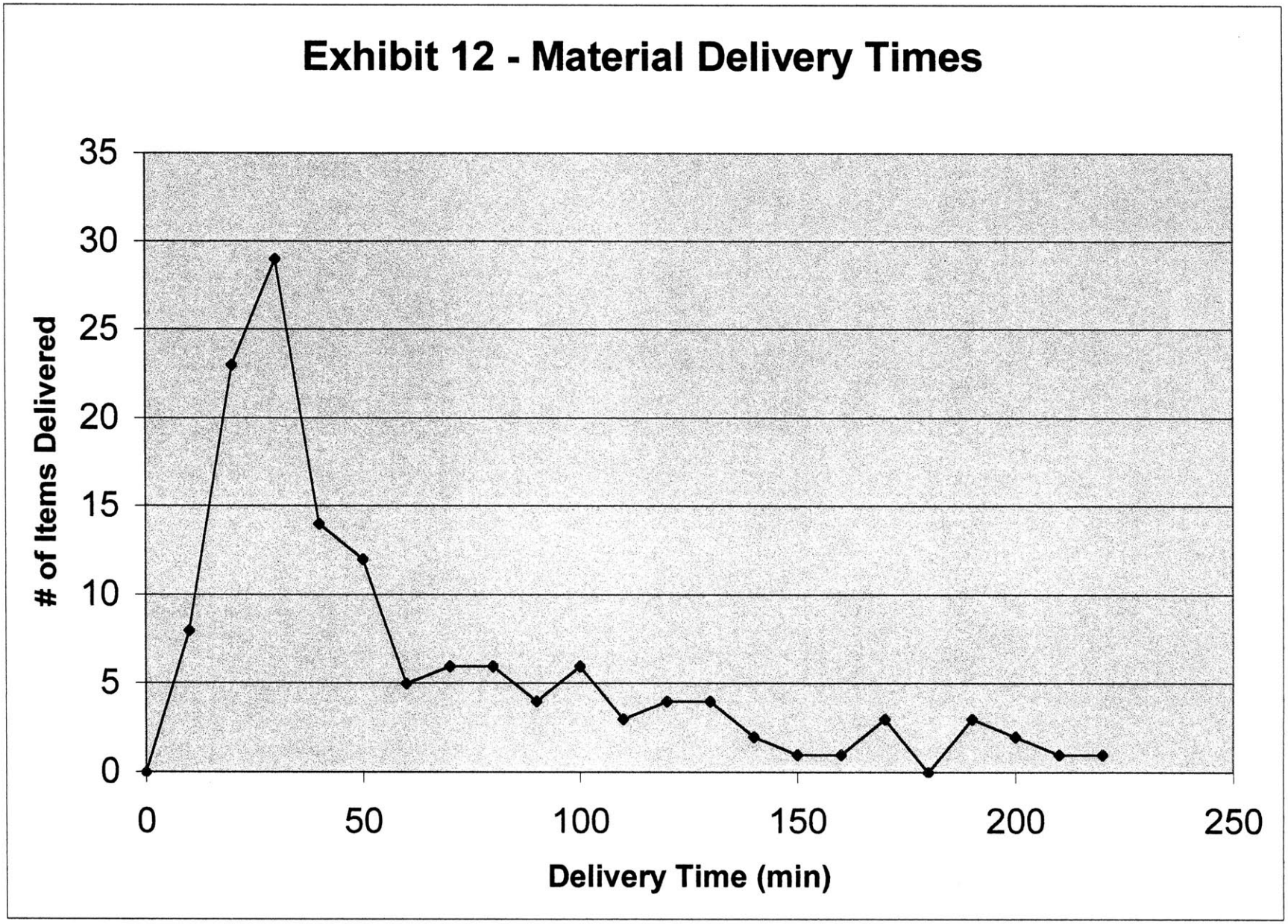


Exhibit 12 - Material Delivery Times
# of Items Delivered
35
30
25
20
15
10
5
0
0
50
100
150
200
250
Delivery Time (min)

Exhibit 13
Long Material Search Periods

	Material Misplacement Rate					
	5%	10%	15%	20%	25%	30%
Recoveries longer than 1 hour	32	51	74	107	167	191
Recoveries longer than 2 hours	9	15	26	32	65	83
Recoveries longer than 3 hours	0	2	5	6	15	26
Total Misplaced Items	75	150	225	300	375	450

Appendix 1:

Simulation Model Logic

Time Units: Minutes
Distance Units: Feet

Locations

Name	Cap	Units	Stats	Rules	Cost
Arrival	1000	1	Time Series	Oldest, ,	
Incoming	100	1	Time Series	Oldest, ,	
Hold	20	1	Time Series	Oldest, ,	
Pickup	20	1	Time Series	Oldest, ,	
Up1	50	1	Time Series	Oldest, ,	
Up2	50	1	Time Series	Oldest, ,	
Up3	50	1	Time Series	Oldest, ,	
Up4	50	1	Time Series	Oldest, ,	
Up5	50	1	Time Series	Oldest, ,	
Up6	50	1	Time Series	Oldest, ,	
Up7	50	1	Time Series	Oldest, ,	
Up8	50	1	Time Series	Oldest, ,	
Up9	50	1	Time Series	Oldest, ,	
Up10	50	1	Time Series	Oldest, ,	
Up11	50	1	Time Series	Oldest, ,	
Up12	50	1	Time Series	Oldest, ,	
Up13	50	1	Time Series	Oldest, ,	
Up14	50	1	Time Series	Oldest, ,	
Up15	50	1	Time Series	Oldest, ,	
Up16	50	1	Time Series	Oldest, ,	
Up17	50	1	Time Series	Oldest, ,	
Up18	50	1	Time Series	Oldest, ,	
Up19	50	1	Time Series	Oldest, ,	
Up20	50	1	Time Series	Oldest, ,	
Down1	100	1	Time Series	Oldest, ,	
Down2	100	1	Time Series	Oldest, ,	
Down3	100	1	Time Series	Oldest, ,	
Down4	100	1	Time Series	Oldest, ,	
Down5	100	1	Time Series	Oldest, ,	
Down6	100	1	Time Series	Oldest, ,	
Down7	100	1	Time Series	Oldest, ,	
Down8	100	1	Time Series	Oldest, ,	
Down9	100	1	Time Series	Oldest, ,	
Down10	100	1	Time Series	Oldest, ,	
Down11	100	1	Time Series	Oldest, ,	
Down12	100	1	Time Series	Oldest, ,	
Down13	100	1	Time Series	Oldest, ,	
Down14	100	1	Time Series	Oldest, ,	
Down15	100	1	Time Series	Oldest, ,	
Down16	100	1	Time Series	Oldest, ,	
Down17	100	1	Time Series	Oldest, ,	
Down18	100	1	Time Series	Oldest, ,	
Down19	100	1	Time Series	Oldest, ,	
Down20	100	1	Time Series	Oldest, ,	
Outgoing	100	1	Time Series	Oldest, ,	
ArrivalX	400	1	Time Series	Oldest, ,	
IncomingX	100	1	Time Series	Oldest, ,	
Up1X	50	1	Time Series	Oldest, ,	
Up2X	50	1	Time Series	Oldest, ,	
Up3X	50	1	Time Series	Oldest, ,	
Up4X	50	1	Time Series	Oldest, ,	
Up5X	50	1	Time Series	Oldest, ,	
Up6X	50	1	Time Series	Oldest, ,	
Up7X	50	1	Time Series	Oldest, ,	
Up8X	50	1	Time Series	Oldest, ,	
Up9X	50	1	Time Series	Oldest, ,	
Up10X	50	1	Time Series	Oldest, ,	
Up11X	50	1	Time Series	Oldest, ,	
Up12X	50	1	Time Series	Oldest, ,	
Up13X	50	1	Time Series	Oldest, ,	
Up14X	50	1	Time Series	Oldest, ,	
Up15X	50	1	Time Series	Oldest, ,	
Up16X	50	1	Time Series	Oldest, ,	
Down1X	100	1	Time Series	Oldest, ,	
Down2X	100	1	Time Series	Oldest, ,	
Down3X	100	1	Time Series	Oldest, ,	

Down4X	100	1	Time Series Oldest, ,
Down5X	100	1	Time Series Oldest, ,
Down6X	100	1	Time Series Oldest, ,
Down7X	100	1	Time Series Oldest, ,
Down8X	100	1	Time Series Oldest, ,
Down9X	100	1	Time Series Oldest, ,
Down10X	100	1	Time Series Oldest, ,
Down11X	100	1	Time Series Oldest, ,
Down12X	100	1	Time Series Oldest, ,
Down13X	100	1	Time Series Oldest, ,
Down14X	100	1	Time Series Oldest, ,
Down15X	100	1	Time Series Oldest, ,
Down16X	100	1	Time Series Oldest, ,
Group1X	100	1	Time Series Oldest, ,
Group2X	100	1	Time Series Oldest, ,
Group3X	100	1	Time Series Oldest, ,
Group4X	100	1	Time Series Oldest, ,
Group5X	100	1	Time Series Oldest, ,
Group6X	100	1	Time Series Oldest, ,
Group7X	100	1	Time Series Oldest, ,
Group8X	100	1	Time Series Oldest, ,
Group9X	100	1	Time Series Oldest, ,
Group10X	100	1	Time Series Oldest, ,
Group11X	100	1	Time Series Oldest, ,
Group12X	100	1	Time Series Oldest, ,
Group13X	100	1	Time Series Oldest, ,
Group14X	100	1	Time Series Oldest, ,
Group15X	100	1	Time Series Oldest, ,
Group16X	100	1	Time Series Oldest, ,

****

* Entities *

****

Name	Speed (fpm)	Stats	Cost
Box_Pallet	880	Time Series	
Positive	880	Time Series	
Negative	880	Time Series	
Truckload	880	Time Series	
Box_PalletX	880	Time Series	
PositiveX	880	Time Series	
NegativeX	880	Time Series	
TruckloadX	880	Time Series	
Truck	880	Time Series	

****

* Path Networks *

****

Name	Type	T/S	From	To	BI	Dist/Time	Speed Factor
Net1	Passing	Speed & Distance	N1	N2	Bi	200	1
		N2	N3	Bi	200	1	
		N3	N4	Bi	350	1	
		N4	N5	Bi	75	1	
		N5	N6	Bi	75	1	
		N4	N7	Bi	75	1	
		N7	N8	Bi	75	1	
		N8	N9	Bi	50	1	
		N7	N10	Bi	50	1	
		N5	N11	Bi	50	1	
		N6	N12	Bi	50	1	
		N3	N13	Bi	450	1	
		N14	N15	Bi	50	1	
		N14	N16	Bi	200	1	
		N17	N18	Bi	300	1	
		N18	N19	Bi	50	1	
		N17	N20	Bi	50	1	
		N14	N21	Bi	200	1	
		N21	N17	Bi	450	1	
		N21	N22	Bi	250	1	
		N22	N23	Bi	50	1	
		N22	N24	Bi	750	1	

N24	N25	Bi	50	1
N25	N26	Bi	50	1
N24	N27	Bi	50	1
N25	N28	Bi	50	1
N26	N29	Bi	100	1
N29	N30	Bi	100	1
N30	N31	Bi	100	1
N31	N32	Bi	100	1
N32	N33	Bi	100	1
N33	N34	Bi	100	1
N34	N35	Bi	100	1
N35	N14	Bi	200	1
N35	N13	Bi	100	1
N34	N36	Bi	50	1
N33	N37	Bi	50	1
N32	N38	Bi	50	1
N31	N39	Bi	50	1
N30	N40	Bi	50	1
N29	N41	Bi	50	1
N42	N43	Bi	350	1
N13	N44	Bi	100	1
N44	N45	Bi	100	1
N45	N46	Bi	100	1
N46	N47	Bi	100	1
N47	N48	Bi	100	1
N48	N49	Bi	100	1
N49	N50	Bi	100	1
N50	N51	Bi	100	1
N51	N52	Bi	250	1
N52	N53	Bi	150	1
N52	N42	Bi	150	1
N53	N54	Bi	100	1
N54	N55	Bi	100	1
N55	N56	Bi	100	1
N56	N57	Bi	100	1
N57	N58	Bi	100	1
N58	N59	Bi	150	1
N59	N60	Bi	100	1
N60	N61	Bi	100	1
N62	N63	Bi	300	1
N63	N64	Bi	50	1
N64	N65	Bi	50	1
N63	N66	Bi	50	1
N62	N67	Bi	50	1
N58	N68	Bi	150	1
N68	N62	Bi	50	1
N61	N69	Bi	50	1
N60	N70	Bi	50	1
N59	N71	Bi	50	1
N57	N72	Bi	50	1
N56	N73	Bi	50	1
N55	N74	Bi	50	1
N54	N75	Bi	50	1
N53	N76	Bi	50	1
N57	N77	Bi	50	1
N55	N78	Bi	50	1
N53	N79	Bi	50	1
N68	N80	Bi	100	1
N80	N81	Bi	100	1
N81	N82	Bi	100	1
N82	N83	Bi	100	1
N83	N84	Bi	100	1
N84	N42	Bi	100	1
N72	N80	Bi	50	1
N73	N81	Bi	50	1
N74	N82	Bi	50	1
N75	N83	Bi	50	1
N76	N84	Bi	50	1
N26	N85	Bi	50	1
N85	N42	Bi	200	1
N85	N86	Bi	50	1
N86	N51	Bi	50	1
N41	N50	Bi	50	1
N40	N49	Bi	50	1
N39	N48	Bi	50	1
N38	N47	Bi	50	1

		N51	N87	Bi	50	1	
		N49	N88	Bi	50	1	
		N37	N46	Bi	50	1	
		N36	N45	Bi	50	1	
		N47	N89	Bi	50	1	
		N44	N90	Bi	50	1	
Net2	Passing	Speed & Distance	N1	N2	Uni	450	1
		N2	N3	Uni	250	1	
		N3	N4	Uni	250	1	
		N4	N5	Uni	250	1	
		N5	N6	Uni	300	1	
		N6	N7	Uni	200	1	
		N7	N8	Uni	200	1	
		N8	N9	Uni	200	1	
		N9	N10	Uni	100	1	
		N10	N11	Uni	100	1	
		N11	N12	Uni	1200	1	
		N12	N13	Uni	100	1	
		N13	N14	Uni	350	1	
		N14	N15	Uni	1050	1	
		N15	N16	Uni	50	1	
		N16	N17	Uni	600	1	
		N17	N18	Uni	500	1	
		N18	N19	Uni	200	1	
		N19	N20	Uni	650	1	
		N20	N21	Uni	250	1	
		N21	N22	Uni	850	1	
		N22	N1	Uni	200	1	

****

* Interfaces *

****

Net	Node	Location
Net1	N1	Incoming
	N2	IncomingX
	N2	Group13X
	N2	Group14X
	N2	Group15X
	N2	Group16X
	N2	Group9X
	N2	Group10X
	N2	Group11X
	N2	Group12X
	N2	Group5X
	N2	Group6X
	N2	Group7X
	N2	Group8X
	N2	Group1X
	N2	Group2X
	N2	Group3X
	N2	Group4X
	N9	Up16X
	N9	Down16X
	N10	Up15X
	N10	Down15X
	N11	Up14X
	N11	Down14X
	N12	Up13X
	N12	Down13X
	N90	Down1
	N89	Down2
	N88	Down3
	N87	Down4
	N87	Up4
	N88	Up3
	N89	Up2
	N90	Up1
	N36	Up1X
	N36	Down1X
	N37	Up2X
	N37	Down2X
	N38	Up3X
	N38	Down3X

	N39	Up4X
	N39	Down4X
	N40	Up5X
	N40	Down5X
	N41	Up6X
	N41	Down6X
	N86	Up7X
	N86	Down7X
	N79	Up5
	N79	Down5
	N78	Up6
	N78	Down6
	N77	Up7
	N77	Down7
	N72	Up12X
	N72	Down12X
	N73	Up11X
	N73	Down11X
	N74	Up10X
	N74	Down10X
	N75	Up9X
	N75	Down9X
	N76	Up8X
	N76	Down8X
	N69	Up10
	N69	Down10
	N70	Up9
	N70	Down9
	N71	Up8
	N71	Down8
	N67	Up11
	N67	Down11
	N66	Up12
	N66	Down12
	N65	Up13
	N65	Down13
	N43	Outgoing
	N16	Up17
	N16	Down17
	N23	Up16
	N23	Down16
	N27	Up15
	N27	Down15
	N28	Up14
	N28	Down14
	N15	Up18
	N15	Down18
	N20	Up19
	N20	Down19
	N19	Up20
	N19	Down20
Net2	N1	Pickup
	N2	Up1
	N3	Up2
	N4	Up3
	N5	Up4
	N6	Up5
	N7	Up6
	N8	Up7
	N9	Up8
	N10	Up9
	N11	Up10
	N12	Up13
	N13	Up12
	N14	Up11
	N15	Up14
	N16	Up15
	N17	Up16
	N18	Up17
	N19	Up18
	N20	Up19
	N21	Up20

****

* Resources *

```
*************************************************************************

                  Res     Ent
Name     Units Stats   Search   Search Path     Motion       Cost
-------- ----- ------- -------- ------------ ------------ ----------
OutOp1   1     By Unit Closest  Oldest Net1    Empty: 880 fpm
                                Home: N43  Full: 880 fpm

OutOp2   1     By Unit Closest  Oldest Net1    Empty: 880 fpm
                                Home: N43  Full: 880 fpm

OutOp3   1     By Unit Closest  Oldest Net1    Empty: 880 fpm
                                Home: N43  Full: 880 fpm

OutOp4   1     By Unit Closest  Oldest Net1    Empty: 880 fpm
                                Home: N43  Full: 880 fpm

OutOp5   1     By Unit Closest  Oldest Net1    Empty: 880 fpm
                                Home: N43  Full: 880 fpm

InOpX    1     By Unit Closest  Oldest Net1    Empty: 880 fpm
                                Home: N2   Full: 880 fpm
                                (Return)

OutOpX1  1     By Unit Closest  Oldest Net1    Empty: 880 fpm
                                Home: N43  Full: 880 fpm

OutOpX2  1     By Unit Closest  Oldest Net1    Empty: 880 fpm
                                Home: N43  Full: 880 fpm

OutOpX3  1     By Unit Closest  Oldest Net1    Empty: 880 fpm
                                Home: N43  Full: 880 fpm

OutOpX4  1     By Unit Closest  Oldest Net1    Empty: 880 fpm
                                Home: N43  Full: 880 fpm

OutOpX5  1     By Unit Closest  Oldest Net1    Empty: 880 fpm
                                Home: N43  Full: 880 fpm

Driver   1     By Unit Closest  Oldest Net2    Empty: 880 fpm
                                Home: N1   Full: 880 fpm

Count    1     By Unit Least Used Oldest      Empty: 150 fpm
                                Full: 150 fpm

CountX   1     By Unit Least Used Oldest      Empty: 150 fpm
                                Full: 150 fpm


*************************************************************************
*                        Processing                                     *
*************************************************************************

                        Process                 Routing

Entity     Location  Operation        Blk  Output     Destination Rule   Move Logic
---------- --------- ---------------  ---  ---------- ---------- ----   ----------
Box_Pallet Arrival   WAIT Artime       1   Box_Pallet Incoming   FIRST 1
Box_Pallet Incoming  GET Count
                     Xout[X,4] = CLOCK(MIN)
                     WAIT N(1,0.30)
                     Xout[X,5] = CLOCK(MIN)
                     FREE Count
                     Counter = 1
                     Select = 0
                     Temp = 0
                     AsdV = AsdA
                     DesV = DesA
                     px = X
                     DO
                      BEGIN
                       Select = Search[AsdV,Counter]
                       Temp = Select + 1
                            IF Select > DesV THEN
                            BEGIN
                                    ROUTE Temp
```

```
      END
      IF Select < DesV THEN
      BEGIN
            ROUTE Temp
      END
 Inc Counter
 END
UNTIL Select = DesV
ROUTE 1          1   Box_Pallet  Hold      FIRST 1
            2*  Negative   Down1     FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3

            3*  Negative   Down2     FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3

            4*  Negative   Down3     FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3

            5*  Negative   Down4     FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3

            6*  Negative   Down5     FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3

            7*  Negative   Down6     FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3

            8*  Negative   Down7     FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3

            9*  Negative   Down8     FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3

           10*  Negative   Down9     FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3

           11*  Negative   Down10    FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3

           12*  Negative   Down11    FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3

           13*  Negative   Down12    FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3

           14*  Negative   Down13    FIRST 1  WAITN = WAITTIME
                                    PUN = PUA
                                    parent = px
                                    INC counter3
```

```
15* Negative   Down14     FIRST 1  WAITN = WAITTIME
                          PUN = PUA
                          parent = px
                          INC counter3

16* Negative   Down15     FIRST 1  WAITN = WAITTIME
                          PUN = PUA
                          parent = px
                          INC counter3

17* Negative   Down16     FIRST 1  WAITN = WAITTIME
                          PUN = PUA
                          parent = px
                          INC counter3

18* Negative   Down17     FIRST 1  WAITN = WAITTIME
                          PUN = PUA
                          parent = px
                          INC counter3

19* Negative   Down18     FIRST 1  WAITN = WAITTIME
                          PUN = PUA
                          parent = px
                          INC counter3

20* Negative   Down19     FIRST 1  WAITN = WAITTIME
                          PUN = PUA
                          parent = px
                          INC counter3

21* Negative   Down20     FIRST 1  WAITN = WAITTIME
                          PUN = PUA
                          parent = px
                          INC counter3

Box_Pallet Hold          1  Box_Pallet Pickup     LOAD 1  IF DesA = 1 THEN
                                                          BEGIN
                                                                INC Q1
                                                          END
                                                    IF DesA = 2 THEN
                                                          BEGIN
                                                                INC Q2
                                                          END
                                                    IF DesA = 3 THEN
                                                          BEGIN
                                                                INC Q3
                                                          END
                                                    IF DesA = 4 THEN
                                                          BEGIN
                                                                INC Q4
                                                          END
                                                    IF DesA = 5 THEN
                                                          BEGIN
                                                                INC Q5
                                                          END
                                                    IF DesA = 6 THEN
                                                          BEGIN
                                                                INC Q6
                                                          END
                                                    IF DesA = 7 THEN
                                                          BEGIN
                                                                INC Q7
                                                          END
                                                    IF DesA = 8 THEN
                                                          BEGIN
                                                                INC Q8
                                                          END
                                                    IF DesA = 9 THEN
                                                          BEGIN
                                                                INC Q9
                                                          END
                                                    IF DesA = 10 THEN
                                                          BEGIN
                                                                INC Q10
                                                          END
```

```
                                        IF DesA = 11 THEN
                                                BEGIN
                                                        INC Q11
                                                END
                                        IF DesA = 12 THEN
                                                BEGIN
                                                        INC Q12
                                                END
                                        IF DesA = 13 THEN
                                                BEGIN
                                                        INC Q13
                                                END
                                        IF DesA = 14 THEN
                                                BEGIN
                                                        INC Q14
                                                END
                                        IF DesA = 15 THEN
                                                BEGIN
                                                        INC Q15
                                                END
                                        IF DesA = 16 THEN
                                                BEGIN
                                                        INC Q16
                                                END
                                        IF DesA = 17 THEN
                                                BEGIN
                                                        INC Q17
                                                END
                                        IF DesA = 18 THEN
                                                BEGIN
                                                        INC Q18
                                                END
                                        IF DesA = 19 THEN
                                                BEGIN
                                                        INC Q19
                                                END
                                        IF DesA = 20 THEN
                                                BEGIN
                                                        INC Q20
                                                END

Box_Pallet Up1      Xout[X,6] = CLOCK(MIN)
           WAIT WAITTIME
           Xout[X,7] = CLOCK(MIN)
                    1   Box_Pallet Outgoing   FIRST 1  searchtime = N(3,1) + N(1,0.5) * C1

                                        IF PUA = 1 THEN
                                        BEGIN
                                                GET OutOp1
                                                Xout[X,8] = CLOCK(MIN)
                                                WAIT searchtime
                                                Xout[X,9] = CLOCK(MIN)
                                                MOVE WITH OutOp1 THEN FREE
                                        END
                                        IF PUA = 2 THEN
                                        BEGIN
                                                GET OutOp2
                                                Xout[X,8] = CLOCK(MIN)
                                                WAIT searchtime
                                                Xout[X,9] = CLOCK(MIN)
                                                MOVE WITH OutOp2 THEN FREE
                                        END
                                        IF PUA = 3 THEN
                                        BEGIN
                                                GET OutOp3
                                                Xout[X,8] = CLOCK(MIN)
                                                WAIT searchtime
                                                Xout[X,9] = CLOCK(MIN)
                                                MOVE WITH OutOp3 THEN FREE
                                        END
                                        IF PUA = 4 THEN
                                        BEGIN
                                                GET OutOp4
                                                Xout[X,8] = CLOCK(MIN)
                                                WAIT searchtime
                                                Xout[X,9] = CLOCK(MIN)
```

```
                                        MOVE WITH OutOp4 THEN FREE
                                    END
                                    IF PUA = 5 THEN
                                    BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                    END

                                    DEC C1

Box_Pallet Up2      Xout[X,6] = CLOCK(MIN)
           WAIT WAITTIME
           Xout[X,7] = CLOCK(MIN)
                    1  Box_Pallet Outgoing  FIRST 1  searchtime = N(3,1) + N(1,0.5) * C2

                                    IF PUA = 1 THEN
                                    BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                    END
                                    IF PUA = 2 THEN
                                    BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                    END
                                    IF PUA = 3 THEN
                                    BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                    END
                                    IF PUA = 4 THEN
                                    BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                    END
                                    IF PUA = 5 THEN
                                    BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                    END

                                    DEC C2

Box_Pallet Up3      Xout[X,6] = CLOCK(MIN)
           WAIT WAITTIME
           Xout[X,7] = CLOCK(MIN)
                    1  Box_Pallet Outgoing  FIRST 1  searchtime = N(3,1) + N(1,0.5) * C3

                                    IF PUA = 1 THEN
                                    BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                    END
                                    IF PUA = 2 THEN
                                    BEGIN
```

```
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                END
                                IF PUA = 3 THEN
                                BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                END
                                IF PUA = 4 THEN
                                BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C3

Box_Pallet Up4      Xout[X,6] = CLOCK(MIN)
            WAIT WAITTIME
            Xout[X,7] = CLOCK(MIN)
                        1   Box_Pallet Outgoing   FIRST 1  searchtime = N(3,1) + N(1,0.5) * C4

                                IF PUA = 1 THEN
                                BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                END
                                IF PUA = 2 THEN
                                BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                END
                                IF PUA = 3 THEN
                                BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                END
                                IF PUA = 4 THEN
                                BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
```

```
                                        MOVE WITH OutOp5 THEN FREE
                                    END

                                    DEC C4

Box_Pallet Up5      Xout[X,6] = CLOCK(MIN)
           WAIT WAITTIME
           Xout[X,7] = CLOCK(MIN)
                    1  Box_Pallet Outgoing  FIRST 1  searchtime = N(3,1) + N(1,0.5) * C5

                                    IF PUA = 1 THEN
                                    BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                    END
                                    IF PUA = 2 THEN
                                    BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                    END
                                    IF PUA = 3 THEN
                                    BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                    END
                                    IF PUA = 4 THEN
                                    BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                    END
                                    IF PUA = 5 THEN
                                    BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                    END

                                    DEC C5

Box_Pallet Up6      Xout[X,6] = CLOCK(MIN)
           WAIT WAITTIME
           Xout[X,7] = CLOCK(MIN)
                    1  Box_Pallet Outgoing  FIRST 1  searchtime = N(3,1) + N(1,0.5) * C6

                                    IF PUA = 1 THEN
                                    BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                    END
                                    IF PUA = 2 THEN
                                    BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                    END
                                    IF PUA = 3 THEN
                                    BEGIN
```

```
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                END
                                IF PUA = 4 THEN
                                BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C6

Box_Pallet Up7    Xout[X,6] = CLOCK(MIN)
        WAIT WAITTIME
        Xout[X,7] = CLOCK(MIN)
                1  Box_Pallet Outgoing  FIRST 1  searchtime = N(3,1) + N(1,0.5) * C7

                                IF PUA = 1 THEN
                                BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                END
                                IF PUA = 2 THEN
                                BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                END
                                IF PUA = 3 THEN
                                BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                END
                                IF PUA = 4 THEN
                                BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C7

Box_Pallet Up8    Xout[X,6] = CLOCK(MIN)
        WAIT WAITTIME
        Xout[X,7] = CLOCK(MIN)
```

```
1  Box_Pallet Outgoing  FIRST 1 searchtime = N(3,1) + N(1,0.5) * C8

                    IF PUA = 1 THEN
                    BEGIN
                            GET OutOp1
                            Xout[X,8] = CLOCK(MIN)
                            WAIT searchtime
                            Xout[X,9] = CLOCK(MIN)
                            MOVE WITH OutOp1 THEN FREE
                    END
                    IF PUA = 2 THEN
                    BEGIN
                            GET OutOp2
                            Xout[X,8] = CLOCK(MIN)
                            WAIT searchtime
                            Xout[X,9] = CLOCK(MIN)
                            MOVE WITH OutOp2 THEN FREE
                    END
                    IF PUA = 3 THEN
                    BEGIN
                            GET OutOp3
                            Xout[X,8] = CLOCK(MIN)
                            WAIT searchtime
                            Xout[X,9] = CLOCK(MIN)
                            MOVE WITH OutOp3 THEN FREE
                    END
                    IF PUA = 4 THEN
                    BEGIN
                            GET OutOp4
                            Xout[X,8] = CLOCK(MIN)
                            WAIT searchtime
                            Xout[X,9] = CLOCK(MIN)
                            MOVE WITH OutOp4 THEN FREE
                    END
                    IF PUA = 5 THEN
                    BEGIN
                            GET OutOp5
                            Xout[X,8] = CLOCK(MIN)
                            WAIT searchtime
                            Xout[X,9] = CLOCK(MIN)
                            MOVE WITH OutOp5 THEN FREE
                    END

                    DEC C8

Box_Pallet Up9    Xout[X,6] = CLOCK(MIN)
           WAIT WAITTIME
           Xout[X,7] = CLOCK(MIN)
                1  Box_Pallet Outgoing  FIRST 1 searchtime = N(3,1) + N(1,0.5) * C9

                    IF PUA = 1 THEN
                    BEGIN
                            GET OutOp1
                            Xout[X,8] = CLOCK(MIN)
                            WAIT searchtime
                            Xout[X,9] = CLOCK(MIN)
                            MOVE WITH OutOp1 THEN FREE
                    END
                    IF PUA = 2 THEN
                    BEGIN
                            GET OutOp2
                            Xout[X,8] = CLOCK(MIN)
                            WAIT searchtime
                            Xout[X,9] = CLOCK(MIN)
                            MOVE WITH OutOp2 THEN FREE
                    END
                    IF PUA = 3 THEN
                    BEGIN
                            GET OutOp3
                            Xout[X,8] = CLOCK(MIN)
                            WAIT searchtime
                            Xout[X,9] = CLOCK(MIN)
                            MOVE WITH OutOp3 THEN FREE
                    END
                    IF PUA = 4 THEN
                    BEGIN
```

```
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C9

Box_Pallet Up10   Xout[X,6] = CLOCK(MIN)
          WAIT WAITTIME
          Xout[X,7] = CLOCK(MIN)
                    1  Box_Pallet Outgoing  FIRST 1  searchtime = N(3,1) + N(1,0.5) * C10

                                IF PUA = 1 THEN
                                BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                END
                                IF PUA = 2 THEN
                                BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                END
                                IF PUA = 3 THEN
                                BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                END
                                IF PUA = 4 THEN
                                BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C10

Box_Pallet Up11   Xout[X,6] = CLOCK(MIN)
          WAIT WAITTIME
          Xout[X,7] = CLOCK(MIN)
                    1  Box_Pallet Outgoing  FIRST 1  searchtime = N(3,1) + N(1,0.5) * C11

                                IF PUA = 1 THEN
                                BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
```

```
                                        MOVE WITH OutOp1 THEN FREE
                                    END
                                    IF PUA = 2 THEN
                                    BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                    END
                                    IF PUA = 3 THEN
                                    BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                    END
                                    IF PUA = 4 THEN
                                    BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                    END
                                    IF PUA = 5 THEN
                                    BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                    END

                                    DEC C11

Box_Pallet Up12    Xout[X,6] = CLOCK(MIN)
        WAIT WAITTIME
        Xout[X,7] = CLOCK(MIN)
                1  Box_Pallet Outgoing  FIRST 1  searchtime = N(3,1) + N(1,0.5) * C12

                                    IF PUA = 1 THEN
                                    BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                    END
                                    IF PUA = 2 THEN
                                    BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                    END
                                    IF PUA = 3 THEN
                                    BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                    END
                                    IF PUA = 4 THEN
                                    BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                    END
                                    IF PUA = 5 THEN
                                    BEGIN
```

```
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C12

Box_Pallet Up13     Xout[X,6] = CLOCK(MIN)
           WAIT WAITTIME
           Xout[X,7] = CLOCK(MIN)
                    1   Box_Pallet Outgoing   FIRST 1  searchtime = N(3,1) + N(1,0.5) * C13

                                IF PUA = 1 THEN
                                BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                END
                                IF PUA = 2 THEN
                                BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                END
                                IF PUA = 3 THEN
                                BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                END
                                IF PUA = 4 THEN
                                BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C13

Box_Pallet Up14     Xout[X,6] = CLOCK(MIN)
           WAIT WAITTIME
           Xout[X,7] = CLOCK(MIN)
                    1   Box_Pallet Outgoing   FIRST 1  searchtime = N(3,1) + N(1,0.5) * C14

                                IF PUA = 1 THEN
                                BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                END
                                IF PUA = 2 THEN
                                BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
```

```
                                        MOVE WITH OutOp2 THEN FREE
                                END
                                IF PUA = 3 THEN
                                BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                END
                                IF PUA = 4 THEN
                                BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C14

Box_Pallet Up15     Xout[X,6] = CLOCK(MIN)
           WAIT WAITTIME
           Xout[X,7] = CLOCK(MIN)
                    1   Box_Pallet Outgoing   FIRST 1  searchtime = N(3,1) + N(1,0.5) * C15

                                IF PUA = 1 THEN
                                BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                END
                                IF PUA = 2 THEN
                                BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                END
                                IF PUA = 3 THEN
                                BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                END
                                IF PUA = 4 THEN
                                BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C15
```

```
Box_Pallet Up16     Xout[X,6] = CLOCK(MIN)
            WAIT WAITTIME
            Xout[X,7] = CLOCK(MIN)
                        1   Box_Pallet Outgoing   FIRST 1  searchtime = N(3,1) + N(1,0.5) * C16

                                                IF PUA = 1 THEN
                                                BEGIN
                                                        GET OutOp1
                                                        Xout[X,8] = CLOCK(MIN)
                                                        WAIT searchtime
                                                        Xout[X,9] = CLOCK(MIN)
                                                        MOVE WITH OutOp1 THEN FREE
                                                END
                                                IF PUA = 2 THEN
                                                BEGIN
                                                        GET OutOp2
                                                        Xout[X,8] = CLOCK(MIN)
                                                        WAIT searchtime
                                                        Xout[X,9] = CLOCK(MIN)
                                                        MOVE WITH OutOp2 THEN FREE
                                                END
                                                IF PUA = 3 THEN
                                                BEGIN
                                                        GET OutOp3
                                                        Xout[X,8] = CLOCK(MIN)
                                                        WAIT searchtime
                                                        Xout[X,9] = CLOCK(MIN)
                                                        MOVE WITH OutOp3 THEN FREE
                                                END
                                                IF PUA = 4 THEN
                                                BEGIN
                                                        GET OutOp4
                                                        Xout[X,8] = CLOCK(MIN)
                                                        WAIT searchtime
                                                        Xout[X,9] = CLOCK(MIN)
                                                        MOVE WITH OutOp4 THEN FREE
                                                END
                                                IF PUA = 5 THEN
                                                BEGIN
                                                        GET OutOp5
                                                        Xout[X,8] = CLOCK(MIN)
                                                        WAIT searchtime
                                                        Xout[X,9] = CLOCK(MIN)
                                                        MOVE WITH OutOp5 THEN FREE
                                                END

                                                DEC C16

Box_Pallet Up17     Xout[X,6] = CLOCK(MIN)
            WAIT WAITTIME
            Xout[X,7] = CLOCK(MIN)
                        1   Box_Pallet Outgoing   FIRST 1  searchtime = N(3,1) + N(1,0.5) * C17

                                                IF PUA = 1 THEN
                                                BEGIN
                                                        GET OutOp1
                                                        Xout[X,8] = CLOCK(MIN)
                                                        WAIT searchtime
                                                        Xout[X,9] = CLOCK(MIN)
                                                        MOVE WITH OutOp1 THEN FREE
                                                END
                                                IF PUA = 2 THEN
                                                BEGIN
                                                        GET OutOp2
                                                        Xout[X,8] = CLOCK(MIN)
                                                        WAIT searchtime
                                                        Xout[X,9] = CLOCK(MIN)
                                                        MOVE WITH OutOp2 THEN FREE
                                                END
                                                IF PUA = 3 THEN
                                                BEGIN
                                                        GET OutOp3
                                                        Xout[X,8] = CLOCK(MIN)
                                                        WAIT searchtime
                                                        Xout[X,9] = CLOCK(MIN)
```

```
                                        MOVE WITH OutOp3 THEN FREE
                                END
                                IF PUA = 4 THEN
                                BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C17

Box_Pallet Up18     Xout[X,6] = CLOCK(MIN)
           WAIT WAITTIME
           Xout[X,7] = CLOCK(MIN)
                    1   Box_Pallet Outgoing   FIRST 1  searchtime = N(3,1) + N(1,0.5) * C18

                                IF PUA = 1 THEN
                                BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                END
                                IF PUA = 2 THEN
                                BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                END
                                IF PUA = 3 THEN
                                BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                END
                                IF PUA = 4 THEN
                                BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C18

Box_Pallet Up19     Xout[X,6] = CLOCK(MIN)
           WAIT WAITTIME
           Xout[X,7] = CLOCK(MIN)
                    1   Box_Pallet Outgoing   FIRST 1  searchtime = N(3,1) + N(1,0.5) * C19

                                IF PUA = 1 THEN
                                BEGIN
```

```
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                END
                                IF PUA = 2 THEN
                                BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                END
                                IF PUA = 3 THEN
                                BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                END
                                IF PUA = 4 THEN
                                BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C19

Box_Pallet Up20     Xout[X,6] = CLOCK(MIN)
           WAIT WAITTIME
           Xout[X,7] = CLOCK(MIN)
                    1   Box_Pallet Outgoing   FIRST 1  searchtime = N(3,1) + N(1,0.5) * C20

                                IF PUA = 1 THEN
                                BEGIN
                                        GET OutOp1
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp1 THEN FREE
                                END
                                IF PUA = 2 THEN
                                BEGIN
                                        GET OutOp2
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp2 THEN FREE
                                END
                                IF PUA = 3 THEN
                                BEGIN
                                        GET OutOp3
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp3 THEN FREE
                                END
                                IF PUA = 4 THEN
                                BEGIN
                                        GET OutOp4
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
```

```
                                        MOVE WITH OutOp4 THEN FREE
                                END
                                IF PUA = 5 THEN
                                BEGIN
                                        GET OutOp5
                                        Xout[X,8] = CLOCK(MIN)
                                        WAIT searchtime
                                        Xout[X,9] = CLOCK(MIN)
                                        MOVE WITH OutOp5 THEN FREE
                                END

                                DEC C20

Negative  Down1   WAIT WAITN
          searchtime = N(3,1) + N(1,0.5) * C1

          IF PUN = 1 THEN
          BEGIN
                GET OutOp1
                verytemp[parent,1] = CLOCK( MIN)
                WAIT searchtime
                verytemp[parent,21] = CLOCK(MIN)
                FREE OutOp1
          END
          IF PUN = 2 THEN
          BEGIN
                GET OutOp2
                verytemp[parent,1] = CLOCK( MIN)
                WAIT searchtime
                verytemp[parent,21] = CLOCK(MIN)
                FREE OutOp2
          END
          IF PUN = 3 THEN
          BEGIN
                GET OutOp3
                verytemp[parent,1] = CLOCK( MIN)
                WAIT searchtime
                verytemp[parent,21] = CLOCK(MIN)
                FREE OutOp3
          END
          IF PUN = 4 THEN
          BEGIN
                GET OutOp4
                verytemp[parent,1] = CLOCK( MIN)
                WAIT searchtime
                verytemp[parent,21] = CLOCK(MIN)
                FREE OutOp4
          END
          IF PUN = 5 THEN
          BEGIN
                GET OutOp5
                verytemp[parent,1] = CLOCK( MIN)
                WAIT searchtime
                verytemp[parent,21] = CLOCK(MIN)
                FREE OutOp5
          END

                        1  Negative  EXIT     FIRST 1
Negative  Down2   WAIT WAITN
          searchtime = N(3,1) + N(1,0.5) * C2

          IF PUN = 1 THEN
          BEGIN
                GET OutOp1
                verytemp[parent,2] = CLOCK(MIN)
                WAIT searchtime
                verytemp[parent,22] = CLOCK(MIN)
                FREE OutOp1
          END
          IF PUN = 2 THEN
          BEGIN
                GET OutOp2
                verytemp[parent,2] = CLOCK(MIN)
                WAIT searchtime
                verytemp[parent,22] = CLOCK(MIN)
                FREE OutOp2
```

```
        END
        IF PUN = 3 THEN
        BEGIN
              GET OutOp3
              verytemp[parent,2] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,22] = CLOCK(MIN)
              FREE OutOp3
        END
        IF PUN = 4 THEN
        BEGIN
              GET OutOp4
              verytemp[parent,2] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,22] = CLOCK(MIN)
              FREE OutOp4
        END
        IF PUN = 5 THEN
        BEGIN
              GET OutOp5
              verytemp[parent,2] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,22] = CLOCK(MIN)
              FREE OutOp5
        END

                    1  Negative  EXIT     FIRST 1
Negative  Down3   WAIT WAITN
        searchtime = N(3,1) + N(1,0.5) * C3

        IF PUN = 1 THEN
        BEGIN
              GET OutOp1
              verytemp[parent,3] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,23] = CLOCK(MIN)
              FREE OutOp1
        END
        IF PUN = 2 THEN
        BEGIN
              GET OutOp2
              verytemp[parent,3] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,23] = CLOCK(MIN)
              FREE OutOp2
        END
        IF PUN = 3 THEN
        BEGIN
              GET OutOp3
              verytemp[parent,3] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,23] = CLOCK(MIN)
              FREE OutOp3
        END
        IF PUN = 4 THEN
        BEGIN
              GET OutOp4
              verytemp[parent,3] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,23] = CLOCK(MIN)
              FREE OutOp4
        END
        IF PUN = 5 THEN
        BEGIN
              GET OutOp5
              verytemp[parent,3] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,23] = CLOCK(MIN)
              FREE OutOp5
        END

                    1  Negative  EXIT     FIRST 1
Negative  Down4   WAIT WAITN
        searchtime = N(3,1) + N(1,0.5) * C4

        IF PUN = 1 THEN
```

```
        BEGIN
              GET OutOp1
              verytemp[parent,4] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,24] = CLOCK(MIN)
              FREE OutOp1
        END
        IF PUN = 2 THEN
        BEGIN
              GET OutOp2
              verytemp[parent,4] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,24] = CLOCK(MIN)
              FREE OutOp2
        END
        IF PUN = 3 THEN
        BEGIN
              GET OutOp3
              verytemp[parent,4] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,24] = CLOCK(MIN)
              FREE OutOp3
        END
        IF PUN = 4 THEN
        BEGIN
              GET OutOp4
              verytemp[parent,4] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,24] = CLOCK(MIN)
              FREE OutOp4
        END
        IF PUN = 5 THEN
        BEGIN
              GET OutOp5
              verytemp[parent,4] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,24] = CLOCK(MIN)
              FREE OutOp5
        END

                    1  Negative  EXIT      FIRST 1
Negative  Down5   WAIT WAITN
        searchtime = N(3,1) + N(1,0.5) * C5

        IF PUN = 1 THEN
        BEGIN
              GET OutOp1
              verytemp[parent,5] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,25] = CLOCK(MIN)
              FREE OutOp1
        END
        IF PUN = 2 THEN
        BEGIN
              GET OutOp2
              verytemp[parent,5] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,25] = CLOCK(MIN)
              FREE OutOp2
        END
        IF PUN = 3 THEN
        BEGIN
              GET OutOp3
              verytemp[parent,5] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,25] = CLOCK(MIN)
              FREE OutOp3
        END
        IF PUN = 4 THEN
        BEGIN
              GET OutOp4
              verytemp[parent,5] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,25] = CLOCK(MIN)
              FREE OutOp4
        END
```

```
        IF PUN = 5 THEN
        BEGIN
              GET OutOp5
              verytemp[parent,5] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,25] = CLOCK(MIN)
              FREE OutOp5
        END

                        1   Negative   EXIT      FIRST 1
Negative   Down6    WAIT WAITN
        searchtime = N(3,1) + N(1,0.5) * C6

        IF PUN = 1 THEN
        BEGIN
              GET OutOp1
              verytemp[parent,6] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,26] = CLOCK(MIN)
              FREE OutOp1
        END
        IF PUN = 2 THEN
        BEGIN
              GET OutOp2
              verytemp[parent,6] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,26] = CLOCK(MIN)
              FREE OutOp2
        END
        IF PUN = 3 THEN
        BEGIN
              GET OutOp3
              verytemp[parent,6] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,26] = CLOCK(MIN)
              FREE OutOp3
        END
        IF PUN = 4 THEN
        BEGIN
              GET OutOp4
              verytemp[parent,6] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,26] = CLOCK(MIN)
              FREE OutOp4
        END
        IF PUN = 5 THEN
        BEGIN
              GET OutOp5
              verytemp[parent,6] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,26] = CLOCK(MIN)
              FREE OutOp5
        END

                        1   Negative   EXIT      FIRST 1
Negative   Down7    WAIT WAITN
        searchtime = N(3,1) + N(1,0.5) * C7

        IF PUN = 1 THEN
        BEGIN
              GET OutOp1
              verytemp[parent,7] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,27] = CLOCK(MIN)
              FREE OutOp1
        END
        IF PUN = 2 THEN
        BEGIN
              GET OutOp2
              verytemp[parent,7] = CLOCK(MIN)
              WAIT searchtime
              verytemp[parent,27] = CLOCK(MIN)
              FREE OutOp2
        END
        IF PUN = 3 THEN
        BEGIN
```

```
            GET OutOp3
            verytemp[parent,7] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,27] = CLOCK(MIN)
            FREE OutOp3
      END
      IF PUN = 4 THEN
      BEGIN
            GET OutOp4
            verytemp[parent,7] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,27] = CLOCK(MIN)
            FREE OutOp4
      END
      IF PUN = 5 THEN
      BEGIN
            GET OutOp5
            verytemp[parent,7] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,27] = CLOCK(MIN)
            FREE OutOp5
      END

                  1  Negative  EXIT     FIRST 1
Negative  Down8   WAIT WAITN
      searchtime = N(3,1) + N(1,0.5) * C8

      IF PUN = 1 THEN
      BEGIN
            GET OutOp1
            verytemp[parent,8] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,28] = CLOCK(MIN)
            FREE OutOp1
      END
      IF PUN = 2 THEN
      BEGIN
            GET OutOp2
            verytemp[parent,8] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,28] = CLOCK(MIN)
            FREE OutOp2
      END
      IF PUN = 3 THEN
      BEGIN
            GET OutOp3
            verytemp[parent,8] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,28] = CLOCK(MIN)
            FREE OutOp3
      END
      IF PUN = 4 THEN
      BEGIN
            GET OutOp4
            verytemp[parent,8] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,28] = CLOCK(MIN)
            FREE OutOp4
      END
      IF PUN = 5 THEN
      BEGIN
            GET OutOp5
            verytemp[parent,8] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,28] = CLOCK(MIN)
            FREE OutOp5
      END

                  1  Negative  EXIT     FIRST 1
Negative  Down9   WAIT WAITN
      searchtime = N(3,1) + N(1,0.5) * C9

      IF PUN = 1 THEN
      BEGIN
            GET OutOp1
            verytemp[parent,9] = CLOCK(MIN)
```

```
        WAIT searchtime
        verytemp[parent,29] = CLOCK(MIN)
        FREE OutOp1
    END
    IF PUN = 2 THEN
    BEGIN
        GET OutOp2
        verytemp[parent,9] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,29] = CLOCK(MIN)
        FREE OutOp2
    END
    IF PUN = 3 THEN
    BEGIN
        GET OutOp3
        verytemp[parent,9] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,29] = CLOCK(MIN)
        FREE OutOp3
    END
    IF PUN = 4 THEN
    BEGIN
        GET OutOp4
        verytemp[parent,9] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,29] = CLOCK(MIN)
        FREE OutOp4
    END
    IF PUN = 5 THEN
    BEGIN
        GET OutOp5
        verytemp[parent,9] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,29] = CLOCK(MIN)
        FREE OutOp5
    END

                1   Negative   EXIT      FIRST 1
Negative   Down10   WAIT WAITN
    searchtime = N(3,1) + N(1,0.5) * C10

    IF PUN = 1 THEN
    BEGIN
        GET OutOp1
        verytemp[parent,10] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,30] = CLOCK(MIN)
        FREE OutOp1
    END
    IF PUN = 2 THEN
    BEGIN
        GET OutOp2
        verytemp[parent,10] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,30] = CLOCK(MIN)
        FREE OutOp2
    END
    IF PUN = 3 THEN
    BEGIN
        GET OutOp3
        verytemp[parent,10] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,30] = CLOCK(MIN)
        FREE OutOp3
    END
    IF PUN = 4 THEN
    BEGIN
        GET OutOp4
        verytemp[parent,10] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,30] = CLOCK(MIN)
        FREE OutOp4
    END
    IF PUN = 5 THEN
    BEGIN
        GET OutOp5
```

```
            verytemp[parent,10] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,30] = CLOCK(MIN)
            FREE OutOp5
        END

                    1  Negative  EXIT      FIRST 1
Negative  Down11  WAIT WAITN
        searchtime = N(3,1) + N(1,0.5) * C11

        IF PUN = 1 THEN
        BEGIN
            GET OutOp1
            verytemp[parent,11] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,31] = CLOCK(MIN)
            FREE OutOp1
        END
        IF PUN = 2 THEN
        BEGIN
            GET OutOp2
            verytemp[parent,11] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,31] = CLOCK(MIN)
            FREE OutOp2
        END
        IF PUN = 3 THEN
        BEGIN
            GET OutOp3
            verytemp[parent,11] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,31] = CLOCK(MIN)
            FREE OutOp3
        END
        IF PUN = 4 THEN
        BEGIN
            GET OutOp4
            verytemp[parent,11] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,31] = CLOCK(MIN)
            FREE OutOp4
        END
        IF PUN = 5 THEN
        BEGIN
            GET OutOp5
            verytemp[parent,11] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,31] = CLOCK(MIN)
            FREE OutOp5
        END

                    1  Negative  EXIT      FIRST 1
Negative  Down12  WAIT WAITN
        searchtime = N(3,1) + N(1,0.5) * C12

        IF PUN = 1 THEN
        BEGIN
            GET OutOp1
            verytemp[parent,12] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,32] = CLOCK(MIN)
            FREE OutOp1
        END
        IF PUN = 2 THEN
        BEGIN
            GET OutOp2
            verytemp[parent,12] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,32] = CLOCK(MIN)
            FREE OutOp2
        END
        IF PUN = 3 THEN
        BEGIN
            GET OutOp3
            verytemp[parent,12] = CLOCK(MIN)
            WAIT searchtime
```

```
            verytemp[parent,32] = CLOCK(MIN)
            FREE OutOp3
        END
        IF PUN = 4 THEN
        BEGIN
            GET OutOp4
            verytemp[parent,12] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,32] = CLOCK(MIN)
            FREE OutOp4
        END
        IF PUN = 5 THEN
        BEGIN
            GET OutOp5
            verytemp[parent,12] = CLOCK(MIN)
            WAIT searchtime
            verytemp[parent,32] = CLOCK(MIN)
            FREE OutOp5
        END

                    1  Negative  EXIT     FIRST 1
Negative  Down13  WAIT WAITN
    searchtime = N(3,1) + N(1,0.5) * C13

    IF PUN = 1 THEN
    BEGIN
        GET OutOp1
        verytemp[parent,13] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,33] = CLOCK(MIN)
        FREE OutOp1
    END
    IF PUN = 2 THEN
    BEGIN
        GET OutOp2
        verytemp[parent,13] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,33] = CLOCK(MIN)
        FREE OutOp2
    END
    IF PUN = 3 THEN
    BEGIN
        GET OutOp3
        verytemp[parent,13] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,33] = CLOCK(MIN)
        FREE OutOp3
    END
    IF PUN = 4 THEN
    BEGIN
        GET OutOp4
        verytemp[parent,13] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,33] = CLOCK(MIN)
        FREE OutOp4
    END
    IF PUN = 5 THEN
    BEGIN
        GET OutOp5
        verytemp[parent,13] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,33] = CLOCK(MIN)
        FREE OutOp5
    END

                    1  Negative  EXIT     FIRST 1
Negative  Down14  WAIT WAITN
    searchtime = N(3,1) + N(1,0.5) * C14

    IF PUN = 1 THEN
    BEGIN
        GET OutOp1
        verytemp[parent,14] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,34] = CLOCK(MIN)
        FREE OutOp1
```

```
END
IF PUN = 2 THEN
BEGIN
      GET OutOp2
      verytemp[parent,14] = CLOCK(MIN)
      WAIT searchtime
      verytemp[parent,34] = CLOCK(MIN)
      FREE OutOp2
END
IF PUN = 3 THEN
BEGIN
      GET OutOp3
      verytemp[parent,14] = CLOCK(MIN)
      WAIT searchtime
      verytemp[parent,34] = CLOCK(MIN)
      FREE OutOp3
END
IF PUN = 4 THEN
BEGIN
      GET OutOp4
      verytemp[parent,14] = CLOCK(MIN)
      WAIT searchtime
      verytemp[parent,34] = CLOCK(MIN)
      FREE OutOp4
END
IF PUN = 5 THEN
BEGIN
      GET OutOp5
      verytemp[parent,14] = CLOCK(MIN)
      WAIT searchtime
      verytemp[parent,34] = CLOCK(MIN)
      FREE OutOp5
END

                  1  Negative  EXIT     FIRST 1
Negative  Down15  WAIT WAITN
  searchtime = N(3,1) + N(1,0.5) * C15

  IF PUN = 1 THEN
  BEGIN
      GET OutOp1
      verytemp[parent,15] = CLOCK(MIN)
      WAIT searchtime
      verytemp[parent,35] = CLOCK(MIN)
      FREE OutOp1
  END
  IF PUN = 2 THEN
  BEGIN
      GET OutOp2
      verytemp[parent,15] = CLOCK(MIN)
      WAIT searchtime
      verytemp[parent,35] = CLOCK(MIN)
      FREE OutOp2
  END
  IF PUN = 3 THEN
  BEGIN
      GET OutOp3
      verytemp[parent,15] = CLOCK(MIN)
      WAIT searchtime
      verytemp[parent,35] = CLOCK(MIN)
      FREE OutOp3
  END
  IF PUN = 4 THEN
  BEGIN
      GET OutOp4
      verytemp[parent,15] = CLOCK(MIN)
      WAIT searchtime
      verytemp[parent,35] = CLOCK(MIN)
      FREE OutOp4
  END
  IF PUN = 5 THEN
  BEGIN
      GET OutOp5
      verytemp[parent,15] = CLOCK(MIN)
      WAIT searchtime
      verytemp[parent,35] = CLOCK(MIN)
```

```
        FREE OutOp5
    END

                1  Negative  EXIT     FIRST 1
Negative  Down16  WAIT WAITN
    searchtime = N(3,1) + N(1,0.5) * C16

    IF PUN = 1 THEN
    BEGIN
        GET OutOp1
        verytemp[parent,16] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,36] = CLOCK(MIN)
        FREE OutOp1
    END
    IF PUN = 2 THEN
    BEGIN
        GET OutOp2
        verytemp[parent,16] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,36] = CLOCK(MIN)
        FREE OutOp2
    END
    IF PUN = 3 THEN
    BEGIN
        GET OutOp3
        verytemp[parent,16] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,36] = CLOCK(MIN)
        FREE OutOp3
    END
    IF PUN = 4 THEN
    BEGIN
        GET OutOp4
        verytemp[parent,16] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,36] = CLOCK(MIN)
        FREE OutOp4
    END
    IF PUN = 5 THEN
    BEGIN
        GET OutOp5
        verytemp[parent,16] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,36] = CLOCK(MIN)
        FREE OutOp5
    END

                1  Negative  EXIT     FIRST 1
Negative  Down17  WAIT WAITN
    searchtime = N(3,1) + N(1,0.5) * C17

    IF PUN = 1 THEN
    BEGIN
        GET OutOp1
        verytemp[parent,17] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,37] = CLOCK(MIN)
        FREE OutOp1
    END
    IF PUN = 2 THEN
    BEGIN
        GET OutOp2
        verytemp[parent,17] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,37] = CLOCK(MIN)
        FREE OutOp2
    END
    IF PUN = 3 THEN
    BEGIN
        GET OutOp3
        verytemp[parent,17] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,37] = CLOCK(MIN)
        FREE OutOp3
    END
```

```
            IF PUN = 4 THEN
            BEGIN
                  GET OutOp4
                  verytemp[parent,17] = CLOCK(MIN)
                  WAIT searchtime
                  verytemp[parent,37] = CLOCK(MIN)
                  FREE OutOp4
            END
            IF PUN = 5 THEN
            BEGIN
                  GET OutOp5
                  verytemp[parent,17] = CLOCK(MIN)
                  WAIT searchtime
                  verytemp[parent,37] = CLOCK(MIN)
                  FREE OutOp5
            END

                        1   Negative   EXIT      FIRST 1
Negative   Down18   WAIT WAITN
            searchtime = N(3,1) + N(1,0.5) * C18

            IF PUN = 1 THEN
            BEGIN
                  GET OutOp1
                  verytemp[parent,18] = CLOCK(MIN)
                  WAIT searchtime
                  verytemp[parent,38] = CLOCK(MIN)
                  FREE OutOp1
            END
            IF PUN = 2 THEN
            BEGIN
                  GET OutOp2
                  verytemp[parent,18] = CLOCK(MIN)
                  WAIT searchtime
                  verytemp[parent,38] = CLOCK(MIN)
                  FREE OutOp2
            END
            IF PUN = 3 THEN
            BEGIN
                  GET OutOp3
                  verytemp[parent,18] = CLOCK(MIN)
                  WAIT searchtime
                  verytemp[parent,38] = CLOCK(MIN)
                  FREE OutOp3
            END
            IF PUN = 4 THEN
            BEGIN
                  GET OutOp4
                  verytemp[parent,18] = CLOCK(MIN)
                  WAIT searchtime
                  verytemp[parent,38] = CLOCK(MIN)
                  FREE OutOp4
            END
            IF PUN = 5 THEN
            BEGIN
                  GET OutOp5
                  verytemp[parent,18] = CLOCK(MIN)
                  WAIT searchtime
                  verytemp[parent,38] = CLOCK(MIN)
                  FREE OutOp5
            END

                        1   Negative   EXIT      FIRST 1
 Negative   Down19   WAIT WAITN
            searchtime = N(3,1) + N(1,0.5) * C19

            IF PUN = 1 THEN
            BEGIN
                  GET OutOp1
                  verytemp[parent,19] = CLOCK(MIN)
                  WAIT searchtime
                  verytemp[parent,39] = CLOCK(MIN)
                  FREE OutOp1
            END
            IF PUN = 2 THEN
            BEGIN
```

```
        GET OutOp2
        verytemp[parent,19] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,39] = CLOCK(MIN)
        FREE OutOp2
    END
    IF PUN = 3 THEN
    BEGIN
        GET OutOp3
        verytemp[parent,19] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,39] = CLOCK(MIN)
        FREE OutOp3
    END
    IF PUN = 4 THEN
    BEGIN
        GET OutOp4
        verytemp[parent,19] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,39] = CLOCK(MIN)
        FREE OutOp4
    END
    IF PUN = 5 THEN
    BEGIN
        GET OutOp5
        verytemp[parent,10] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,30] = CLOCK(MIN)
        FREE OutOp5
    END

                1  Negative  EXIT     FIRST 1
Negative  Down20  WAIT WAITN
    searchtime = N(3,1) + N(1,0.5) * C20

    IF PUN = 1 THEN
    BEGIN
        GET OutOp1
        verytemp[parent,20] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,40] = CLOCK(MIN)
        FREE OutOp1
    END
    IF PUN = 2 THEN
    BEGIN
        GET OutOp2
        verytemp[parent,20] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,40] = CLOCK(MIN)
        FREE OutOp2
    END
    IF PUN = 3 THEN
    BEGIN
        GET OutOp3
        verytemp[parent,20] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,40] = CLOCK(MIN)
        FREE OutOp3
    END
    IF PUN = 4 THEN
    BEGIN
        GET OutOp4
        verytemp[parent,20] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,40] = CLOCK(MIN)
        FREE OutOp4
    END
    IF PUN = 5 THEN
    BEGIN
        GET OutOp5
        verytemp[parent,20] = CLOCK(MIN)
        WAIT searchtime
        verytemp[parent,40] = CLOCK(MIN)
        FREE OutOp5
    END
```

```
                        1  Negative  EXIT     FIRST 1
Box_Pallet Outgoing Xout[X,10] = CLOCK(MIN)
                        1  Box_Pallet EXIT     FIRST 1
Box_PalletX ArrivalX WAIT ArtimeX

            XoutX[XX,2] = DesAX
            INC counter7X     1  Box_PalletX IncomingX  FIRST 1
Box_PalletX IncomingX GET CountX
            XoutX[XX,4] = CLOCK( MIN)
            WAIT N(1,0.30)
            XoutX[XX,5] = CLOCK( MIN)
            FREE CountX
            CounterX = 1
            SelectX = 0
            TempX = 0
            AsdVX = AsdAX
            DesVX = DesAX
            pxX = XX
            DO
             BEGIN
              SelectX = SearchX[AsdVX,CounterX]
              TempX = SelectX + 16
                  IF SelectX > DesVX THEN
                  BEGIN
                        ROUTE TempX
                  END
                  IF SelectX < DesVX THEN
                  BEGIN
                        ROUTE TempX
                  END
              INC CounterX
             END
            UNTIL SelectX = DesVX
            ROUTE SelectX     1  PositiveX  Group1X   FIRST 1  INC C1X
                        2  PositiveX  Group2X   FIRST 1  INC C2X
                        3  PositiveX  Group3X   FIRST 1  INC C3X
                        4  PositiveX  Group4X   FIRST 1  INC C4X
                        5  PositiveX  Group5X   FIRST 1  INC C5X
                        6  PositiveX  Group6X   FIRST 1  INC C6X
                        7  PositiveX  Group7X   FIRST 1  INC C7X
                        8  PositiveX  Group8X   FIRST 1  INC C8X
                        9  PositiveX  Group9X   FIRST 1  INC C9X
                        10 PositiveX  Group10X  FIRST 1  INC C10X
                        11 PositiveX  Group11X  FIRST 1  INC C11X
                        12 PositiveX  Group12X  FIRST 1  INC C12X
                        13 PositiveX  Group13X  FIRST 1  INC C13X
                        14 PositiveX  Group14X  FIRST 1  INC C14X
                        15 PositiveX  Group15X  FIRST 1  INC C15X
                        16 PositiveX  Group16X  FIRST 1  INC C16X
                        17* NegativeX  Down1X    FIRST 1  WAITNX = WAITTIMEX
                                       PUNX = PUAX
                                       parentX = pxX
                                       INC counter3X
                        18* NegativeX  Down2X    FIRST 1  WAITNX = WAITTIMEX
                                       PUNX = PUAX
                                       parentX = pxX
                                       INC counter3X
                        19* NegativeX  Down3X    FIRST 1  WAITNX = WAITTIMEX
                                       PUNX = PUAX
                                       parentX = pxX
                                       INC counter3X
                        20* NegativeX  Down4X    FIRST 1  WAITNX = WAITTIMEX
                                       PUNX = PUAX
                                       parentX = pxX
                                       INC counter3X
                        21* NegativeX  Down5X    FIRST 1  WAITNX = WAITTIMEX
                                       PUNX = PUAX
                                       parentX = pxX
                                       INC counter3X
                        22* NegativeX  Down6X    FIRST 1  WAITNX = WAITTIMEX
                                       PUNX = PUAX
                                       parentX = pxX
                                       INC counter3X
                        23* NegativeX  Down7X    FIRST 1  WAITNX = WAITTIMEX
                                       PUNX = PUAX
                                       parentX = pxX
```

```
                                INC counter3X
                        24* NegativeX  Down8X    FIRST 1  WAITNX = WAITTIMEX
                                PUNX = PUAX
                                parentX = pxX
                                INC counter3X
                        25* NegativeX  Down9X    FIRST 1  WAITNX = WAITTIMEX
                                PUNX = PUAX
                                parentX = pxX
                                INC counter3X
                        26* NegativeX  Down10X   FIRST 1  WAITNX = WAITTIMEX
                                PUNX = PUAX
                                parentX = pxX
                                INC counter3X
                        27* NegativeX  Down11X   FIRST 1  WAITNX = WAITTIMEX
                                PUNX = PUAX
                                parentX = pxX
                                INC counter3X
                        28* NegativeX  Down12X   FIRST 1  WAITNX = WAITTIMEX
                                PUNX = PUAX
                                parentX = pxX
                                INC counter3X
                        29* NegativeX  Down13X   FIRST 1  WAITNX = WAITTIMEX
                                PUNX = PUAX
                                parentX = pxX
                                INC counter3X
                        30* NegativeX  Down14X   FIRST 1  WAITNX = WAITTIMEX
                                PUNX = PUAX
                                parentX = pxX
                                INC counter3X
                        31* NegativeX  Down15X   FIRST 1  WAITNX = WAITTIMEX
                                PUNX = PUAX
                                parentX = pxX
                                INC counter3X
                        32* NegativeX  Down16X   FIRST 1  WAITNX = WAITTIMEX
                                PUNX = PUAX
                                parentX = pxX
                                INC counter3X
PositiveX  Up1X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                        1  PositiveX  Outgoing  FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C1X

                                IF PUAX = 1 THEN
                                BEGIN
                                        GET OutOpX1
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX1 THEN FREE
                                END
                                IF PUAX = 2 THEN
                                BEGIN
                                        GET OutOpX2
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX2 THEN FREE
                                END
                                IF PUAX = 3 THEN
                                BEGIN
                                        GET OutOpX3
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX3 THEN FREE
                                END
                                IF PUAX = 4 THEN
                                BEGIN
                                        GET OutOpX4
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX4 THEN FREE
                                END
                                IF PUAX = 5 THEN
                                BEGIN
```

```
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX5 THEN FREE
                                END

                                DEC C1X

PositiveX  Up2X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                    1  PositiveX  Outgoing  FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C2X

                                IF PUAX = 1 THEN
                                BEGIN
                                        GET OutOpX1
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX1 THEN FREE
                                END
                                IF PUAX = 2 THEN
                                BEGIN
                                        GET OutOpX2
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX2 THEN FREE
                                END
                                IF PUAX = 3 THEN
                                BEGIN
                                        GET OutOpX3
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX3 THEN FREE
                                END
                                IF PUAX = 4 THEN
                                BEGIN
                                        GET OutOpX4
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX4 THEN FREE
                                END
                                IF PUAX = 5 THEN
                                BEGIN
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX5 THEN FREE
                                END

                                DEC C2X

PositiveX  Up3X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                    1  PositiveX  Outgoing  FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C3X

                                IF PUAX = 1 THEN
                                BEGIN
                                        GET OutOpX1
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX1 THEN FREE
                                END
                                IF PUAX = 2 THEN
                                BEGIN
                                        GET OutOpX2
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
```

```
                                        MOVE WITH OutOpX2 THEN FREE
                                END
                                IF PUAX = 3 THEN
                                BEGIN
                                        GET OutOpX3
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX3 THEN FREE
                                END
                                IF PUAX = 4 THEN
                                BEGIN
                                        GET OutOpX4
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX4 THEN FREE
                                END
                                IF PUAX = 5 THEN
                                BEGIN
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX5 THEN FREE
                                END

                                DEC C3X

PositiveX  Up4X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                    1  PositiveX  Outgoing  FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C4X

                                IF PUAX = 1 THEN
                                BEGIN
                                        GET OutOpX1
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX1 THEN FREE
                                END
                                IF PUAX = 2 THEN
                                BEGIN
                                        GET OutOpX2
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX2 THEN FREE
                                END
                                IF PUAX = 3 THEN
                                BEGIN
                                        GET OutOpX3
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX3 THEN FREE
                                END
                                IF PUAX = 4 THEN
                                BEGIN
                                        GET OutOpX4
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX4 THEN FREE
                                END
                                IF PUAX = 5 THEN
                                BEGIN
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX5 THEN FREE
                                END

                                DEC C4X
```

```
PositiveX  Up5X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                    1  PositiveX  Outgoing  FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C5X

                                        IF PUAX = 1 THEN
                                        BEGIN
                                                GET OutOpX1
                                                XoutX[XX,8] = CLOCK( MIN)
                                                WAIT searchtimeX
                                                XoutX[XX,9] = CLOCK( MIN)
                                                MOVE WITH OutOpX1 THEN FREE
                                        END
                                        IF PUAX = 2 THEN
                                        BEGIN
                                                GET OutOpX2
                                                XoutX[XX,8] = CLOCK( MIN)
                                                WAIT searchtimeX
                                                XoutX[XX,9] = CLOCK( MIN)
                                                MOVE WITH OutOpX2 THEN FREE
                                        END
                                        IF PUAX = 3 THEN
                                        BEGIN
                                                GET OutOpX3
                                                XoutX[XX,8] = CLOCK( MIN)
                                                WAIT searchtimeX
                                                XoutX[XX,9] = CLOCK( MIN)
                                                MOVE WITH OutOpX3 THEN FREE
                                        END
                                        IF PUAX = 4 THEN
                                        BEGIN
                                                GET OutOpX4
                                                XoutX[XX,8] = CLOCK( MIN)
                                                WAIT searchtimeX
                                                XoutX[XX,9] = CLOCK( MIN)
                                                MOVE WITH OutOpX4 THEN FREE
                                        END
                                        IF PUAX = 5 THEN
                                        BEGIN
                                                GET OutOpX5
                                                XoutX[XX,8] = CLOCK( MIN)
                                                WAIT searchtimeX
                                                XoutX[XX,9] = CLOCK( MIN)
                                                MOVE WITH OutOpX5 THEN FREE
                                        END

                                        DEC C5X

PositiveX  Up6X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                    1  PositiveX  Outgoing  FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C6X

                                        IF PUAX = 1 THEN
                                        BEGIN
                                                GET OutOpX1
                                                XoutX[XX,8] = CLOCK( MIN)
                                                WAIT searchtimeX
                                                XoutX[XX,9] = CLOCK( MIN)
                                                MOVE WITH OutOpX1 THEN FREE
                                        END
                                        IF PUAX = 2 THEN
                                        BEGIN
                                                GET OutOpX2
                                                XoutX[XX,8] = CLOCK( MIN)
                                                WAIT searchtimeX
                                                XoutX[XX,9] = CLOCK( MIN)
                                                MOVE WITH OutOpX2 THEN FREE
                                        END
                                        IF PUAX = 3 THEN
                                        BEGIN
                                                GET OutOpX3
                                                XoutX[XX,8] = CLOCK( MIN)
                                                WAIT searchtimeX
                                                XoutX[XX,9] = CLOCK( MIN)
```

```
                                        MOVE WITH OutOpX3 THEN FREE
                                    END
                                    IF PUAX = 4 THEN
                                    BEGIN
                                        GET OutOpX4
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX4 THEN FREE
                                    END
                                    IF PUAX = 5 THEN
                                    BEGIN
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX5 THEN FREE
                                    END

                                    DEC C6X

PositiveX  Up7X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                    1   PositiveX  Outgoing   FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C7X

                                    IF PUAX = 1 THEN
                                    BEGIN
                                        GET OutOpX1
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX1 THEN FREE
                                    END
                                    IF PUAX = 2 THEN
                                    BEGIN
                                        GET OutOpX2
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX2 THEN FREE
                                    END
                                    IF PUAX = 3 THEN
                                    BEGIN
                                        GET OutOpX3
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX3 THEN FREE
                                    END
                                    IF PUAX = 4 THEN
                                    BEGIN
                                        GET OutOpX4
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX4 THEN FREE
                                    END
                                    IF PUAX = 5 THEN
                                    BEGIN
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX5 THEN FREE
                                    END

                                    DEC C7X

 PositiveX  Up8X    XoutX[XX,6] = CLOCK( MIN)
            WAIT WAITTIMEX
            XoutX[XX,7] = CLOCK( MIN)
                     1   PositiveX  Outgoing   FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C8X

                                    IF PUAX = 1 THEN
                                    BEGIN
```

```
                    GET OutOpX1
                    XoutX[XX,8] = CLOCK( MIN)
                    WAIT searchtimeX
                    XoutX[XX,9] = CLOCK( MIN)
                    MOVE WITH OutOpX1 THEN FREE
                END
                IF PUAX = 2 THEN
                BEGIN
                    GET OutOpX2
                    XoutX[XX,8] = CLOCK( MIN)
                    WAIT searchtimeX
                    XoutX[XX,9] = CLOCK( MIN)
                    MOVE WITH OutOpX2 THEN FREE
                END
                IF PUAX = 3 THEN
                BEGIN
                    GET OutOpX3
                    XoutX[XX,8] = CLOCK( MIN)
                    WAIT searchtimeX
                    XoutX[XX,9] = CLOCK( MIN)
                    MOVE WITH OutOpX3 THEN FREE
                END
                IF PUAX = 4 THEN
                BEGIN
                    GET OutOpX4
                    XoutX[XX,8] = CLOCK( MIN)
                    WAIT searchtimeX
                    XoutX[XX,9] = CLOCK( MIN)
                    MOVE WITH OutOpX4 THEN FREE
                END
                IF PUAX = 5 THEN
                BEGIN
                    GET OutOpX5
                    XoutX[XX,8] = CLOCK( MIN)
                    WAIT searchtimeX
                    XoutX[XX,9] = CLOCK( MIN)
                    MOVE WITH OutOpX5 THEN FREE
                END

                DEC C8X

PositiveX  Up9X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                1  PositiveX  Outgoing  FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C9X

                IF PUAX = 1 THEN
                BEGIN
                    GET OutOpX1
                    XoutX[XX,8] = CLOCK( MIN)
                    WAIT searchtimeX
                    XoutX[XX,9] = CLOCK( MIN)
                    MOVE WITH OutOpX1 THEN FREE
                END
                IF PUAX = 2 THEN
                BEGIN
                    GET OutOpX2
                    XoutX[XX,8] = CLOCK( MIN)
                    WAIT searchtimeX
                    XoutX[XX,9] = CLOCK( MIN)
                    MOVE WITH OutOpX2 THEN FREE
                END
                IF PUAX = 3 THEN
                BEGIN
                    GET OutOpX3
                    XoutX[XX,8] = CLOCK( MIN)
                    WAIT searchtimeX
                    XoutX[XX,9] = CLOCK( MIN)
                    MOVE WITH OutOpX3 THEN FREE
                END
                IF PUAX = 4 THEN
                BEGIN
                    GET OutOpX4
                    XoutX[XX,8] = CLOCK( MIN)
                    WAIT searchtimeX
                    XoutX[XX,9] = CLOCK( MIN)
```

```
                                        MOVE WITH OutOpX4 THEN FREE
                                    END
                                    IF PUAX = 5 THEN
                                    BEGIN
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX5 THEN FREE
                                    END

                                    DEC C9X

PositiveX  Up10X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                    1   PositiveX  Outgoing   FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C10X

                                    IF PUAX = 1 THEN
                                    BEGIN
                                        GET OutOpX1
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX1 THEN FREE
                                    END
                                    IF PUAX = 2 THEN
                                    BEGIN
                                        GET OutOpX2
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX2 THEN FREE
                                    END
                                    IF PUAX = 3 THEN
                                    BEGIN
                                        GET OutOpX3
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX3 THEN FREE
                                    END
                                    IF PUAX = 4 THEN
                                    BEGIN
                                        GET OutOpX4
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX4 THEN FREE
                                    END
                                    IF PUAX = 5 THEN
                                    BEGIN
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX5 THEN FREE
                                    END

                                    DEC C10X

PositiveX  Up11X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                    1   PositiveX  Outgoing   FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C11X

                                    IF PUAX = 1 THEN
                                    BEGIN
                                        GET OutOpX1
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX1 THEN FREE
                                    END
                                    IF PUAX = 2 THEN
                                    BEGIN
```

```
                                        GET OutOpX2
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX2 THEN FREE
                                END
                                IF PUAX = 3 THEN
                                BEGIN
                                        GET OutOpX3
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX3 THEN FREE
                                END
                                IF PUAX = 4 THEN
                                BEGIN
                                        GET OutOpX4
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX4 THEN FREE
                                END
                                IF PUAX = 5 THEN
                                BEGIN
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX5 THEN FREE
                                END

                                DEC C11X

PositiveX  Up12X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                    1   PositiveX  Outgoing   FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C12X

                                IF PUAX = 1 THEN
                                BEGIN
                                        GET OutOpX1
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX1 THEN FREE
                                END
                                IF PUAX = 2 THEN
                                BEGIN
                                        GET OutOpX2
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX2 THEN FREE
                                END
                                IF PUAX = 3 THEN
                                BEGIN
                                        GET OutOpX3
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX3 THEN FREE
                                END
                                IF PUAX = 4 THEN
                                BEGIN
                                        GET OutOpX4
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX4 THEN FREE
                                END
                                IF PUAX = 5 THEN
                                BEGIN
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
```

```
                                        MOVE WITH OutOpX5 THEN FREE
                                    END

                                    DEC C12X

PositiveX  Up13X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                    1   PositiveX  Outgoing   FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C13X

                                    IF PUAX = 1 THEN
                                    BEGIN
                                        GET OutOpX1
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX1 THEN FREE
                                    END
                                    IF PUAX = 2 THEN
                                    BEGIN
                                        GET OutOpX2
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX2 THEN FREE
                                    END
                                    IF PUAX = 3 THEN
                                    BEGIN
                                        GET OutOpX3
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX3 THEN FREE
                                    END
                                    IF PUAX = 4 THEN
                                    BEGIN
                                        GET OutOpX4
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX4 THEN FREE
                                    END
                                    IF PUAX = 5 THEN
                                    BEGIN
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX5 THEN FREE
                                    END

                                    DEC C13X

PositiveX  Up14X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                    1   PositiveX  Outgoing   FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C14X

                                    IF PUAX = 1 THEN
                                    BEGIN
                                        GET OutOpX1
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX1 THEN FREE
                                    END
                                    IF PUAX = 2 THEN
                                    BEGIN
                                        GET OutOpX2
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX2 THEN FREE
                                    END
                                    IF PUAX = 3 THEN
                                    BEGIN
```

```
                                        GET OutOpX3
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX3 THEN FREE
                                END
                                IF PUAX = 4 THEN
                                BEGIN
                                        GET OutOpX4
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX4 THEN FREE
                                END
                                IF PUAX = 5 THEN
                                BEGIN
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX5 THEN FREE
                                END

                                DEC C14X

PositiveX  Up15X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
                    1   PositiveX  Outgoing   FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C15X

                                IF PUAX = 1 THEN
                                BEGIN
                                        GET OutOpX1
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX1 THEN FREE
                                END
                                IF PUAX = 2 THEN
                                BEGIN
                                        GET OutOpX2
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX2 THEN FREE
                                END
                                IF PUAX = 3 THEN
                                BEGIN
                                        GET OutOpX3
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX3 THEN FREE
                                END
                                IF PUAX = 4 THEN
                                BEGIN
                                        GET OutOpX4
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX4 THEN FREE
                                END
                                IF PUAX = 5 THEN
                                BEGIN
                                        GET OutOpX5
                                        XoutX[XX,8] = CLOCK( MIN)
                                        WAIT searchtimeX
                                        XoutX[XX,9] = CLOCK( MIN)
                                        MOVE WITH OutOpX5 THEN FREE
                                END

                                DEC C15X

PositiveX  Up16X    XoutX[XX,6] = CLOCK( MIN)
           WAIT WAITTIMEX
           XoutX[XX,7] = CLOCK( MIN)
```

```
1  PositiveX  Outgoing  FIRST 1  searchtimeX = N(3,1) + N(1,0.5) * C16X

                    IF PUAX = 1 THEN
                    BEGIN
                            GET OutOpX1
                            XoutX[XX,8] = CLOCK( MIN)
                            WAIT searchtimeX
                            XoutX[XX,9] = CLOCK( MIN)
                            MOVE WITH OutOpX1 THEN FREE
                    END
                    IF PUAX = 2 THEN
                    BEGIN
                            GET OutOpX2
                            XoutX[XX,8] = CLOCK( MIN)
                            WAIT searchtimeX
                            XoutX[XX,9] = CLOCK( MIN)
                            MOVE WITH OutOpX2 THEN FREE
                    END
                    IF PUAX = 3 THEN
                    BEGIN
                            GET OutOpX3
                            XoutX[XX,8] = CLOCK( MIN)
                            WAIT searchtimeX
                            XoutX[XX,9] = CLOCK( MIN)
                            MOVE WITH OutOpX3 THEN FREE
                    END
                    IF PUAX = 4 THEN
                    BEGIN
                            GET OutOpX4
                            XoutX[XX,8] = CLOCK( MIN)
                            WAIT searchtimeX
                            XoutX[XX,9] = CLOCK( MIN)
                            MOVE WITH OutOpX4 THEN FREE
                    END
                    IF PUAX = 5 THEN
                    BEGIN
                            GET OutOpX5
                            XoutX[XX,8] = CLOCK( MIN)
                            WAIT searchtimeX
                            XoutX[XX,9] = CLOCK( MIN)
                            MOVE WITH OutOpX5 THEN FREE
                    END

                    DEC C16X

NegativeX  Down1X  WAIT WAITNX
           searchtimeX = N(3,1) + N(1,0.5) * C1X

           IF PUNX = 1 THEN
           BEGIN
                GET OutOpX1
                verytempX[parentX,1] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,17] = CLOCK( MIN)
                FREE OutOpX1
           END
           IF PUNX = 2 THEN
           BEGIN
                GET OutOpX2
                verytempX[parentX,1] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,17] = CLOCK( MIN)
                FREE OutOpX2
           END
           IF PUNX = 3 THEN
           BEGIN
                GET OutOpX3
                verytempX[parentX,1] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,17] = CLOCK( MIN)
                FREE OutOpX3
           END
           IF PUNX = 4 THEN
           BEGIN
                GET OutOpX4
                verytempX[parentX,1] = CLOCK( MIN)
```

```
                WAIT searchtimeX
                verytempX[parentX,17] = CLOCK( MIN)
                FREE OutOpX4
            END
            IF PUNX = 5 THEN
            BEGIN
                GET OutOpX5
                verytempX[parentX,1] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,17] = CLOCK( MIN)
                FREE OutOpX5
            END             1   NegativeX  EXIT      FIRST 1
NegativeX   Down2X   WAIT WAITNX
            searchtimeX = N(3,1) + N(1,0.5) * C2X

            IF PUNX = 1 THEN
            BEGIN
                GET OutOpX1
                verytempX[parentX,2] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,18] = CLOCK( MIN)
                FREE OutOpX1
            END
            IF PUNX = 2 THEN
            BEGIN
                GET OutOpX2
                verytempX[parentX,2] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,18] = CLOCK( MIN)
                FREE OutOpX2
            END
            IF PUNX = 3 THEN
            BEGIN
                GET OutOpX3
                verytempX[parentX,2] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,18] = CLOCK( MIN)
                FREE OutOpX3
            END
            IF PUNX = 4 THEN
            BEGIN
                GET OutOpX4
                verytempX[parentX,2] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,18] = CLOCK( MIN)
                FREE OutOpX4
            END
            IF PUNX = 5 THEN
            BEGIN
                GET OutOpX5
                verytempX[parentX,2] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,18] = CLOCK( MIN)
                FREE OutOpX5
            END             1   NegativeX  EXIT      FIRST 1
NegativeX   Down3X   WAIT WAITNX
            searchtimeX = N(3,1) + N(1,0.5) * C3X

            IF PUNX = 1 THEN
            BEGIN
                GET OutOpX1
                verytempX[parentX,3] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,19] = CLOCK( MIN)
                FREE OutOpX1
            END
            IF PUNX = 2 THEN
            BEGIN
                GET OutOpX2
                verytempX[parentX,3] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,19] = CLOCK( MIN)
                FREE OutOpX2
            END
            IF PUNX = 3 THEN
            BEGIN
```

```
            GET OutOpX3
            verytempX[parentX,3] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,19] = CLOCK( MIN)
            FREE OutOpX3
        END
        IF PUNX = 4 THEN
        BEGIN
            GET OutOpX4
            verytempX[parentX,3] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,19] = CLOCK( MIN)
            FREE OutOpX4
        END
        IF PUNX = 5 THEN
        BEGIN
            GET OutOpX5
            verytempX[parentX,3] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,19] = CLOCK( MIN)
            FREE OutOpX5
        END             1   NegativeX  EXIT      FIRST 1
NegativeX  Down4X   WAIT WAITNX
        searchtimeX = N(3,1) + N(1,0.5) * C4X

        IF PUNX = 1 THEN
        BEGIN
            GET OutOpX1
            verytempX[parentX,4] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,20] = CLOCK( MIN)
            FREE OutOpX1
        END
        IF PUNX = 2 THEN
        BEGIN
            GET OutOpX2
            verytempX[parentX,4] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,20] = CLOCK( MIN)
            FREE OutOpX2
        END
        IF PUNX = 3 THEN
        BEGIN
            GET OutOpX3
            verytempX[parentX,4] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,20] = CLOCK( MIN)
            FREE OutOpX3
        END
        IF PUNX = 4 THEN
        BEGIN
            GET OutOpX4
            verytempX[parentX,4] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,20] = CLOCK( MIN)
            FREE OutOpX4
        END
        IF PUNX = 5 THEN
        BEGIN
            GET OutOpX5
            verytempX[parentX,4] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,20] = CLOCK( MIN)
            FREE OutOpX5
        END             1   NegativeX  EXIT      FIRST 1
NegativeX  Down5X   WAIT WAITNX
        searchtimeX = N(3,1) + N(1,0.5) * C5X

        IF PUNX = 1 THEN
        BEGIN
            GET OutOpX1
            verytempX[parentX,5] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,21] = CLOCK( MIN)
            FREE OutOpX1
        END
```

```
            IF PUNX = 2 THEN
            BEGIN
                  GET OutOpX2
                  verytempX[parentX,5] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,21] = CLOCK( MIN)
                  FREE OutOpX2
            END
            IF PUNX = 3 THEN
            BEGIN
                  GET OutOpX3
                  verytempX[parentX,5] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,21] = CLOCK( MIN)
                  FREE OutOpX3
            END
            IF PUNX = 4 THEN
            BEGIN
                  GET OutOpX4
                  verytempX[parentX,5] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,21] = CLOCK( MIN)
                  FREE OutOpX4
            END
            IF PUNX = 5 THEN
            BEGIN
                  GET OutOpX5
                  verytempX[parentX,5] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,21] = CLOCK( MIN)
                  FREE OutOpX5
            END            1   NegativeX  EXIT      FIRST 1
NegativeX  Down6X   WAIT WAITNX
            searchtimeX = N(3,1) + N(1,0.5) * C6X

            IF PUNX = 1 THEN
            BEGIN
                  GET OutOpX1
                  verytempX[parentX,6] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,22] = CLOCK( MIN)
                  FREE OutOpX1
            END
            IF PUNX = 2 THEN
            BEGIN
                  GET OutOpX2
                  verytempX[parentX,6] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,22] = CLOCK( MIN)
                  FREE OutOpX2
            END
            IF PUNX = 3 THEN
            BEGIN
                  GET OutOpX3
                  verytempX[parentX,6] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,22] = CLOCK( MIN)
                  FREE OutOpX3
            END
            IF PUNX = 4 THEN
            BEGIN
                  GET OutOpX4
                  verytempX[parentX,6] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,22] = CLOCK( MIN)
                  FREE OutOpX4
            END
            IF PUNX = 5 THEN
            BEGIN
                  GET OutOpX5
                  verytempX[parentX,6] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,22] = CLOCK( MIN)
                  FREE OutOpX5
            END            1   NegativeX  EXIT      FIRST 1
NegativeX  Down7X   WAIT WAITNX
```

```
            searchtimeX = N(3,1) + N(1,0.5) * C7X

            IF PUNX = 1 THEN
            BEGIN
                  GET OutOpX1
                  verytempX[parentX,7] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,23] = CLOCK( MIN)
                  FREE OutOpX1
            END
            IF PUNX = 2 THEN
            BEGIN
                  GET OutOpX2
                  verytempX[parentX,7] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,23] = CLOCK( MIN)
                  FREE OutOpX2
            END
            IF PUNX = 3 THEN
            BEGIN
                  GET OutOpX3
                  verytempX[parentX,7] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,23] = CLOCK( MIN)
                  FREE OutOpX3
            END
            IF PUNX = 4 THEN
            BEGIN
                  GET OutOpX4
                  verytempX[parentX,7] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,23] = CLOCK( MIN)
                  FREE OutOpX4
            END
            IF PUNX = 5 THEN
            BEGIN
                  GET OutOpX5
                  verytempX[parentX,7] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,23] = CLOCK( MIN)
                  FREE OutOpX5
            END               1   NegativeX  EXIT      FIRST 1
NegativeX   Down8X   WAIT WAITNX
            searchtimeX = N(3,1) + N(1,0.5) * C8X

            IF PUNX = 1 THEN
            BEGIN
                  GET OutOpX1
                  verytempX[parentX,8] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,24] = CLOCK( MIN)
                  FREE OutOpX1
            END
            IF PUNX = 2 THEN
            BEGIN
                  GET OutOpX2
                  verytempX[parentX,8] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,24] = CLOCK( MIN)
                  FREE OutOpX2
            END
            IF PUNX = 3 THEN
            BEGIN
                  GET OutOpX3
                  verytempX[parentX,8] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,24] = CLOCK( MIN)
                  FREE OutOpX3
            END
            IF PUNX = 4 THEN
            BEGIN
                  GET OutOpX4
                  verytempX[parentX,8] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,24] = CLOCK( MIN)
                  FREE OutOpX4
```

```
            END
            IF PUNX = 5 THEN
            BEGIN
                  GET OutOpX5
                  verytempX[parentX,8] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,24] = CLOCK( MIN)
                  FREE OutOpX5
            END               1   NegativeX  EXIT      FIRST 1
NegativeX   Down9X   WAIT WAITNX
            searchtimeX = N(3,1) + N(1,0.5) * C9X

            IF PUNX = 1 THEN
            BEGIN
                  GET OutOpX1
                  verytempX[parentX,9] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,25] = CLOCK( MIN)
                  FREE OutOpX1
            END
            IF PUNX = 2 THEN
            BEGIN
                  GET OutOpX2
                  verytempX[parentX,9] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,25] = CLOCK( MIN)
                  FREE OutOpX2
            END
            IF PUNX = 3 THEN
            BEGIN
                  GET OutOpX3
                  verytempX[parentX,9] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,25] = CLOCK( MIN)
                  FREE OutOpX3
            END
            IF PUNX = 4 THEN
            BEGIN
                  GET OutOpX4
                  verytempX[parentX,9] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,25] = CLOCK( MIN)
                  FREE OutOpX4
            END
            IF PUNX = 5 THEN
            BEGIN
                  GET OutOpX5
                  verytempX[parentX,9] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,25] = CLOCK( MIN)
                  FREE OutOpX5
            END               1   NegativeX  EXIT      FIRST 1
NegativeX   Down10X  WAIT WAITNX
            searchtimeX = N(3,1) + N(1,0.5) * C10X

            IF PUNX = 1 THEN
            BEGIN
                  GET OutOpX1
                  verytempX[parentX,10] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,26] = CLOCK( MIN)
                  FREE OutOpX1
            END
            IF PUNX = 2 THEN
            BEGIN
                  GET OutOpX2
                  verytempX[parentX,10] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,26] = CLOCK( MIN)
                  FREE OutOpX2
            END
            IF PUNX = 3 THEN
            BEGIN
                  GET OutOpX3
                  verytempX[parentX,10] = CLOCK( MIN)
                  WAIT searchtimeX
```

```
            verytempX[parentX,26] = CLOCK( MIN)
            FREE OutOpX3
        END
        IF PUNX = 4 THEN
        BEGIN
            GET OutOpX4
            verytempX[parentX,10] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,26] = CLOCK( MIN)
            FREE OutOpX4
        END
        IF PUNX = 5 THEN
        BEGIN
            GET OutOpX5
            verytempX[parentX,10] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,26] = CLOCK( MIN)
            FREE OutOpX5
        END          1   NegativeX  EXIT      FIRST 1
NegativeX  Down11X  WAIT WAITNX
        searchtimeX = N(3,1) + N(1,0.5) * C11X

        IF PUNX = 1 THEN
        BEGIN
            GET OutOpX1
            verytempX[parentX,11] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,27] = CLOCK( MIN)
            FREE OutOpX1
        END
        IF PUNX = 2 THEN
        BEGIN
            GET OutOpX2
            verytempX[parentX,11] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,27] = CLOCK( MIN)
            FREE OutOpX2
        END
        IF PUNX = 3 THEN
        BEGIN
            GET OutOpX3
            verytempX[parentX,11] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,27] = CLOCK( MIN)
            FREE OutOpX3
        END
        IF PUNX = 4 THEN
        BEGIN
            GET OutOpX4
            verytempX[parentX,11] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,27] = CLOCK( MIN)
            FREE OutOpX4
        END
        IF PUNX = 5 THEN
        BEGIN
            GET OutOpX5
            verytempX[parentX,11] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,27] = CLOCK( MIN)
            FREE OutOpX5
        END          1   NegativeX  EXIT      FIRST 1
NegativeX  Down12X  WAIT WAITNX
        searchtimeX = N(3,1) + N(1,0.5) * C12X

        IF PUNX = 1 THEN
        BEGIN
            GET OutOpX1
            verytempX[parentX,12] = CLOCK( MIN)
            WAIT searchtimeX
            verytempX[parentX,28] = CLOCK( MIN)
            FREE OutOpX1
        END
        IF PUNX = 2 THEN
        BEGIN
            GET OutOpX2
```

```
                verytempX[parentX,12] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,28] = CLOCK( MIN)
                FREE OutOpX2
        END
        IF PUNX = 3 THEN
        BEGIN
                GET OutOpX3
                verytempX[parentX,12] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,28] = CLOCK( MIN)
                FREE OutOpX3
        END
        IF PUNX = 4 THEN
        BEGIN
                GET OutOpX4
                verytempX[parentX,12] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,28] = CLOCK( MIN)
                FREE OutOpX4
        END
        IF PUNX = 5 THEN
        BEGIN
                GET OutOpX5
                verytempX[parentX,12] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,28] = CLOCK( MIN)
                FREE OutOpX5
        END             1   NegativeX  EXIT      FIRST 1
NegativeX  Down13X  WAIT WAITNX
        searchtimeX = N(3,1) + N(1,0.5) * C13X

        IF PUNX = 1 THEN
        BEGIN
                GET OutOpX1
                verytempX[parentX,13] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,29] = CLOCK( MIN)
                FREE OutOpX1
        END
        IF PUNX = 2 THEN
        BEGIN
                GET OutOpX2
                verytempX[parentX,13] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,29] = CLOCK( MIN)
                FREE OutOpX2
        END
        IF PUNX = 3 THEN
        BEGIN
                GET OutOpX3
                verytempX[parentX,13] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,29] = CLOCK( MIN)
                FREE OutOpX3
        END
        IF PUNX = 4 THEN
        BEGIN
                GET OutOpX4
                verytempX[parentX,13] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,29] = CLOCK( MIN)
                FREE OutOpX4
        END
        IF PUNX = 5 THEN
        BEGIN
                GET OutOpX5
                verytempX[parentX,13] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,29] = CLOCK( MIN)
                FREE OutOpX5
        END             1   NegativeX  EXIT      FIRST 1
NegativeX  Down14X  WAIT WAITNX
        searchtimeX = N(3,1) + N(1,0.5) * C14X

        IF PUNX = 1 THEN
```

```
            BEGIN
                  GET OutOpX1
                  verytempX[parentX,14] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,30] = CLOCK( MIN)
                  FREE OutOpX1
            END
            IF PUNX = 2 THEN
            BEGIN
                  GET OutOpX2
                  verytempX[parentX,14] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,30] = CLOCK( MIN)
                  FREE OutOpX2
            END
            IF PUNX = 3 THEN
            BEGIN
                  GET OutOpX3
                  verytempX[parentX,14] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,30] = CLOCK( MIN)
                  FREE OutOpX3
            END
            IF PUNX = 4 THEN
            BEGIN
                  GET OutOpX4
                  verytempX[parentX,14] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,30] = CLOCK( MIN)
                  FREE OutOpX4
            END
            IF PUNX = 5 THEN
            BEGIN
                  GET OutOpX5
                  verytempX[parentX,14] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,30] = CLOCK( MIN)
                  FREE OutOpX5
            END            1   NegativeX  EXIT      FIRST 1
NegativeX  Down15X  WAIT WAITNX
            searchtimeX = N(3,1) + N(1,0.5) * C15X

            IF PUNX = 1 THEN
            BEGIN
                  GET OutOpX1
                  verytempX[parentX,15] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,31] = CLOCK( MIN)
                  FREE OutOpX1
            END
            IF PUNX = 2 THEN
            BEGIN
                  GET OutOpX2
                  verytempX[parentX,15] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,31] = CLOCK( MIN)
                  FREE OutOpX2
            END
            IF PUNX = 3 THEN
            BEGIN
                  GET OutOpX3
                  verytempX[parentX,15] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,31] = CLOCK( MIN)
                  FREE OutOpX3
            END
            IF PUNX = 4 THEN
            BEGIN
                  GET OutOpX4
                  verytempX[parentX,15] = CLOCK( MIN)
                  WAIT searchtimeX
                  verytempX[parentX,31] = CLOCK( MIN)
                  FREE OutOpX4
            END
            IF PUNX = 5 THEN
            BEGIN
```

```
                GET OutOpX5
                verytempX[parentX,15] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,31] = CLOCK( MIN)
                FREE OutOpX5
            END             1   NegativeX  EXIT      FIRST 1
NegativeX   Down16X  WAIT WAITNX
            searchtimeX = N(3,1) + N(1,0.5) * C16X

            IF PUNX = 1 THEN
            BEGIN
                GET OutOpX1
                verytempX[parentX,16] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,32] = CLOCK( MIN)
                FREE OutOpX1
            END
            IF PUNX = 2 THEN
            BEGIN
                GET OutOpX2
                verytempX[parentX,16] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,32] = CLOCK( MIN)
                FREE OutOpX2
            END
            IF PUNX = 3 THEN
            BEGIN
                GET OutOpX3
                verytempX[parentX,16] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,32] = CLOCK( MIN)
                FREE OutOpX3
            END
            IF PUNX = 4 THEN
            BEGIN
                GET OutOpX4
                verytempX[parentX,16] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,32] = CLOCK( MIN)
                FREE OutOpX4
            END
            IF PUNX = 5 THEN
            BEGIN
                GET OutOpX5
                verytempX[parentX,16] = CLOCK( MIN)
                WAIT searchtimeX
                verytempX[parentX,32] = CLOCK( MIN)
                FREE OutOpX5
            END             1   NegativeX  EXIT      FIRST 1
PositiveX   Outgoing XoutX[XX,10] = CLOCK( MIN)
                        1   PositiveX  EXIT      FIRST 1
PositiveX   Group1X  grpX = QuantityX[counter5X,1]
              GROUP grpX as TruckloadX
PositiveX   Group2X  grpX = QuantityX[counter5X,1]
              GROUP grpX as TruckloadX
PositiveX   Group3X  grpX = QuantityX[counter5X,1]
              GROUP grpX as TruckloadX
PositiveX   Group4X  grpX = QuantityX[counter5X,1]
              GROUP grpX as TruckloadX
PositiveX   Group5X  grpX = QuantityX[counter5X,1]
              GROUP grpX as TruckloadX
PositiveX   Group6X  grpX = QuantityX[counter5X,1]
              GROUP grpX as TruckloadX
PositiveX   Group7X  grpX = QuantityX[counter5X,1]
              GROUP grpX as TruckloadX
PositiveX   Group8X  grpX = QuantityX[counter5X,1]
              GROUP grpX as TruckloadX
PositiveX   Group9X  grpX = QuantityX[counter5X,1]
              GROUP grpX as TruckloadX
PositiveX   Group10X grpX = QuantityX[counter5X,1]
              GROUP grpX as TruckloadX
PositiveX   Group11X grpX = QuantityX[counter5X,1]
              GROUP grpX as TruckloadX
PositiveX   Group12X grpX = QuantityX[counter5X,1]
              GROUP grpX as TruckloadX
PositiveX   Group13X grpX = QuantityX[counter5X,1]
```

```
            GROUP grpX as TruckloadX
PositiveX  Group14X  grpX = QuantityX[counter5X,1]
            GROUP grpX as TruckloadX
PositiveX  Group15X  grpX = QuantityX[counter5X,1]
            GROUP grpX as TruckloadX
PositiveX  Group16X  grpX = QuantityX[counter5X,1]
            GROUP grpX as TruckloadX
TruckloadX Group1X              1  TruckloadX Up1X     FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group2X              1  TruckloadX Up2X     FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group3X              1  TruckloadX Up3X     FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group4X              1  TruckloadX Up4X     FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group5X              1  TruckloadX Up5X     FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group6X              1  TruckloadX Up6X     FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group7X              1  TruckloadX Up7X     FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group8X              1  TruckloadX Up8X     FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group9X              1  TruckloadX Up9X     FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group10X             1  TruckloadX Up10X    FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group11X             1  TruckloadX Up11X    FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group12X             1  TruckloadX Up12X    FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group13X             1  TruckloadX Up13X    FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group14X             1  TruckloadX Up14X    FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group15X             1  TruckloadX Up15X    FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE
TruckloadX Group16X             1  TruckloadX Up16X    FIRST 1 INC counter5X
                                       MOVE WITH InOpX THEN FREE

TruckloadX Up1X    UNGROUP
TruckloadX Up2X    UNGROUP
TruckloadX Up3X    UNGROUP
TruckloadX Up4X    UNGROUP
TruckloadX Up5X    UNGROUP
TruckloadX Up6X    UNGROUP
TruckloadX Up7X    UNGROUP
TruckloadX Up8X    UNGROUP
TruckloadX Up9X    UNGROUP
TruckloadX Up10X   UNGROUP
TruckloadX Up11X   UNGROUP
TruckloadX Up12X   UNGROUP
TruckloadX Up13X   UNGROUP
TruckloadX Up14X   UNGROUP
TruckloadX Up15X   UNGROUP
TruckloadX Up16X   UNGROUP
Truck      Pickup   IF unit2 < 123 THEN
                    BEGIN
                    holdno = Quantity[unit2,1]
                    LOAD holdno
                    ROUTE 1
                    END
                    ELSE
                    BEGIN
                    ROUTE 2
                    END            1  Truck    Up1     FIRST 1 INC unit2
                                                       MOVE WITH Driver THEN FREE

                                   2  Truck    EXIT    FIRST 1
Truck      Up1      UNLOAD Q1
                INC C1, Q1         1  Truck    Up2     FIRST 1 MOVE WITH Driver THEN FREE
Truck      Up2      UNLOAD Q2
                INC C2, Q2         1  Truck    Up3     FIRST 1 MOVE WITH Driver THEN FREE
Truck      Up3      UNLOAD Q3
                INC C3, Q3         1  Truck    Up4     FIRST 1 MOVE WITH Driver THEN FREE
Truck      Up4      UNLOAD Q4
                INC C4, Q4         1  Truck    Up5     FIRST 1 MOVE WITH Driver THEN FREE
```

```
Truck    Up5    UNLOAD Q5
           INC C5, Q5        1  Truck    Up6     FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up6    UNLOAD Q6
           INC C6, Q6        1  Truck    Up7     FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up7    UNLOAD Q7
           INC C7, Q7        1  Truck    Up8     FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up8    UNLOAD Q8
           INC C8, Q8        1  Truck    Up9     FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up9    UNLOAD Q9
           INC C9, Q9        1  Truck    Up10    FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up10   UNLOAD Q10
           INC C10, Q10      1  Truck    Up13    FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up13   UNLOAD Q13
           INC C13, Q13      1  Truck    Up12    FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up12   UNLOAD Q12
           INC C12, Q12      1  Truck    Up11    FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up11   UNLOAD Q11
           INC C11, Q11      1  Truck    Up14    FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up14   UNLOAD Q14
           INC C14, Q14      1  Truck    Up15    FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up15   UNLOAD Q15
           INC C15, Q15      1  Truck    Up16    FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up16   UNLOAD Q16
           INC C16, Q16      1  Truck    Up17    FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up17   UNLOAD Q17
           INC C17, Q17      1  Truck    Up18    FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up18   UNLOAD Q18
           INC C18, Q18      1  Truck    Up19    FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up19   UNLOAD Q19
           INC C19, Q19      1  Truck    Up20    FIRST 1  MOVE WITH Driver THEN FREE
Truck    Up20   UNLOAD Q20
           INC C20, Q20      1  Truck    Pickup  FIRST 1  Q1 = 0
                                                          Q2 = 0
                                                          Q3 = 0
                                                          Q4 = 0
                                                          Q5 = 0
                                                          Q6 = 0
                                                          Q7 = 0
                                                          Q8 = 0
                                                          Q9 = 0
                                                          Q10 = 0
                                                          Q11 = 0
                                                          Q12 = 0
                                                          Q13 = 0
                                                          Q14 = 0
                                                          Q15 = 0
                                                          Q16 = 0
                                                          Q17 = 0
                                                          Q18 = 0
                                                          Q19 = 0
                                                          Q20 = 0
                                                          MOVE WITH Driver THEN FREE

*****************************************************************************
*                          Arrivals                                         *
*****************************************************************************

 Entity      Location Qty each  First Time Occurrences Frequency  Logic
 ----------- -------- --------  ---------- ----------- ---------  ------------
 Box_Pallet  Arrival  1         0          1000        0.01       AsdA = Assign[Unit,1]
                                                                  Quan = Assign[Unit,3]
                                                                  GQ = Assign[Unit,4]
                                                                  Artime = Assign[Unit,5]
                                                                  DesA = Assign[Unit,2]
                                                                  X = Unit

                                                                  WAITTIME = N(1000,300)
                                                                  WorkT = WAITTIME
                                                                  PUA = PU()

                                                                  Xout[X,1] = AsdA
                                                                  Xout[X,2] = DesA
                                                                  Xout[X,3] = WAITTIME
                                                                  INC Unit
 Box_PalletX ArrivalX 1         15         400         0.01       AsdAX = AssignX[UnitX,1]
```

```
DesAX = AssignX[UnitX,2]
QuanX = AssignX[UnitX,3]
GQX = AssignX[UnitX,3]
ArtimeX = AssignX[UnitX,5]
XX = UnitX


WAITTIMEX = N(400,100)
WorkTX = WAITTIMEX
PUAX = PUX()
INC UnitX

XoutX[XX,3] = WAITTIMEX
XoutX[XX,1] = AsdAX
```

Truck Pickup 1 0 1 1

****

* Attributes *

****

ID	Type	Classification
AsdA	Integer	Entity
DesA	Integer	Entity
Quan	Integer	Entity
GQ	Integer	Entity
Artime	Integer	Entity
WAITTIME	Real	Entity
WAITN	Real	Entity
AsdAX	Integer	Entity
DesAX	Integer	Entity
QuanX	Integer	Entity
GQX	Integer	Entity
ArtimeX	Integer	Entity
WAITTIMEX	Real	Entity
WAITNX	Real	Entity
PUA	Integer	Entity
PUAX	Integer	Entity
PUN	Integer	Entity
PUNX	Integer	Entity
X	Integer	Entity
XX	Integer	Entity
parent	Integer	Entity
parentX	Integer	Entity

****

* Variables (global) *

****

ID	Type	Initial value	Stats
AsdV	Integer	0	Time Series
DesV	Integer	0	Time Series
Counter	Integer	0	Time Series
Select	Integer	0	Time Series
Temp	Integer	0	Time Series
Unit	Integer	1	Time Series
unit2	Integer	1	Time Series
C1	Integer	0	Time Series
C2	Integer	0	Time Series
C3	Integer	0	Time Series
C4	Integer	0	Time Series
C5	Integer	0	Time Series
C6	Integer	0	Time Series
C7	Integer	0	Time Series
C8	Integer	0	Time Series
C9	Integer	0	Time Series
C10	Integer	0	Time Series
C11	Integer	0	Time Series
C12	Integer	0	Time Series
C13	Integer	0	Time Series

C14	Integer	0	Time Series
C15	Integer	0	Time Series
C16	Integer	0	Time Series
C17	Integer	0	Time Series
C18	Integer	0	Time Series
C19	Integer	0	Time Series
C20	Integer	0	Time Series
counter2	Integer	1	Time Series
xxx	Integer	0	Time Series
WorkT	Real	0	Time Series
counter3	Integer	1	Time Series
counter4	Integer	1	Time Series
counter5	Integer	1	Time Series
grp	Integer	0	Time Series
holdno	Integer	0	Time Series
AsdVX	Integer	0	Time Series
DesVX	Integer	0	Time Series
CounterX	Integer	0	Time Series
SelectX	Integer	0	Time Series
TempX	Integer	0	Time Series
UnitX	Integer	1	Time Series
C1X	Integer	0	Time Series
C2X	Integer	0	Time Series
C3X	Integer	0	Time Series
C4X	Integer	0	Time Series
C5X	Integer	0	Time Series
C6X	Integer	0	Time Series
C7X	Integer	0	Time Series
C8X	Integer	0	Time Series
C9X	Integer	0	Time Series
C10X	Integer	0	Time Series
C11X	Integer	0	Time Series
C12X	Integer	0	Time Series
C13X	Integer	0	Time Series
C14X	Integer	0	Time Series
C15X	Integer	0	Time Series
C16X	Integer	0	Time Series
counter2X	Integer	1	Time Series
xxxX	Integer	0	Time Series
WorkTX	Real	0	Time Series
counter3X	Integer	1	Time Series
counter4X	Integer	1	Time Series
counter5X	Integer	1	Time Series
counter7X	Integer	1	Time Series
grpx	Integer	0	Time Series
Q1	Integer	0	Time Series
Q2	Integer	0	Time Series
Q3	Integer	0	Time Series
Q4	Integer	0	Time Series
Q5	Integer	0	Time Series
Q6	Integer	0	Time Series
Q7	Integer	0	Time Series
Q8	Integer	0	Time Series
Q9	Integer	0	Time Series
Q10	Integer	0	Time Series
Q11	Integer	0	Time Series
Q12	Integer	0	Time Series
Q13	Integer	0	Time Series
Q14	Integer	0	Time Series
Q15	Integer	0	Time Series
Q16	Integer	0	Time Series
Q17	Integer	0	Time Series
Q18	Integer	0	Time Series
Q19	Integer	0	Time Series
Q20	Integer	0	Time Series
Searchtime	Real	0	Time Series
SearchtimeX	Real	0	Time Series
px	Integer	0	Time Series
pxX	Integer	0	Time Series

****
* Arrays *
****

ID	Dimensions	Type