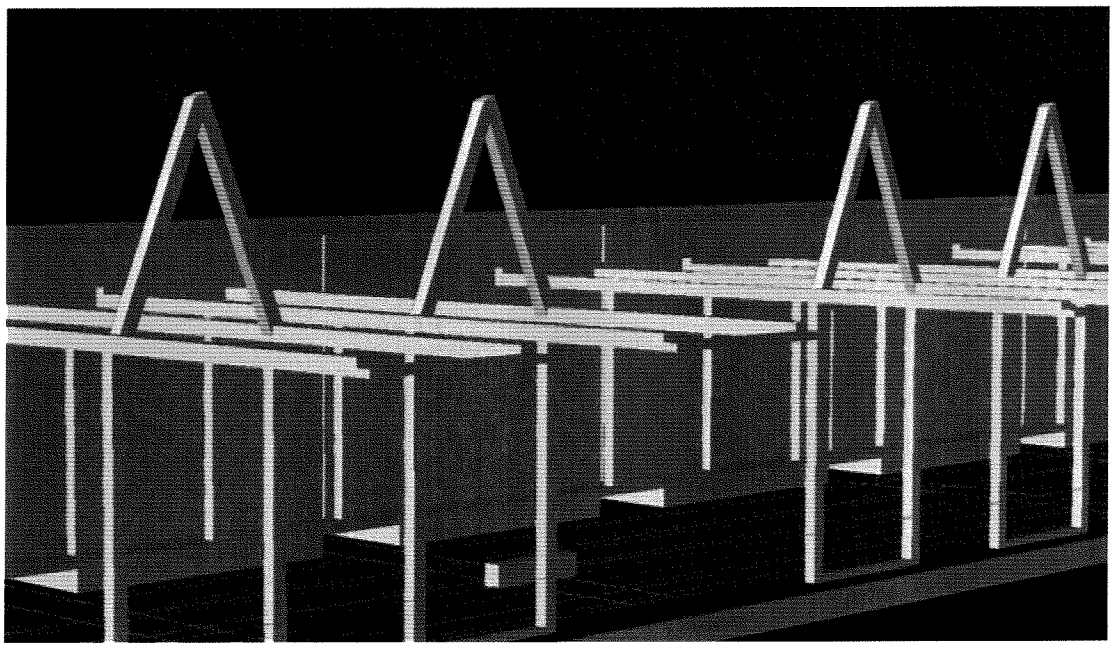
Study on the Design and Simulation of a High capacity transhipment terminal

- ***Report***



Impression of the portainers and container warehouse

December 2000

Hein van Hees
Thies Visscher

TUDelft
Delft University of Technology

Faculty of Civil Engineering and Geosciences
Department of Hydraulic and Geotechnical Engineering
Section of Hydraulic Engineering

Foreword

This report concerns the study on the design and simulation of a high capacity container terminal. It has been carried out as a Master of Science thesis for section Ports and Terminals of the faculty of Civil Engineering and Geosciences at the Delft University of Technology. The appendices of this report have been divided into two separate volumes.

We would like to thank prof. ir. H. Ligteringen and ir. R. Groenveld of the section Ports and Waterways and prof. ir. J.C. Rijsenbrij of the section Transport technology of the faculty of Mechanical Engineering and Marine Technology for their support and critical notes. We would like to thank prof. Hoong Chor Chin for the efforts he made as this study has been partly carried out at the National University of Singapore.

Finally we would like to thank our parents for their support, and especially Willem Visscher for his enthusiastically given advises.

Hein van Hees
Thies Visscher
Delft, December 2000

Contents

Foreword

Summary

List of Figures

List of tables

List of Definitions

1 Introduction **1**
2 Problem analysis **5**
2.1 Objective *5*
2.2 Problem approach *5*
3 Assumptions and boundary conditions concerning the terminal **7**
3.1 Containers *7*
3.2 The vessels *8*
3.2.1 The Mega vessel 8
3.2.2 The feeders 9
3.3 The berths *9*
3.4 Handling strategies of the portainers *10*
3.5 Terminal strategies *11*
3.6 Arrival cycles of the vessels *15*
3.7 Required container storage capacity *16*
4 Choice of the terminal components and their capacities **17**
4.1 Portainers *17*
4.2 Container storage system *21*
4.2.1 Description of the chosen warehousing variant 22
4.2.2 Adaptations to the chosen warehousing concept 25
4.3 Horizontal transportation *28*
4.3.1 Rubber tired automated guided vehicles (agv's) 29
4.3.2 Rail guided vehicles (rgv's) 30
4.3.3 Criteria and choice 31
5 The design of the terminal Layout **33**
5.1 Positioning of the container warehouses *33*
5.2 Positioning and capacity of the rgv lanes *34*
5.2.1 Inter-berth lanes 34
5.2.2 Yard-lanes 36
5.3 Additional terminal design aspects *38*
5.4 Handling strategies applied to the terminal Layout *40*
5.4.1 The unloading of a container vessel 40
5.4.2 The adjacent loading of a mega vessel 41
5.4.3 The adjacent loading of a feeder vessel 42
5.4.4 The shuffle process on the terminal 42
5.5 Number of warehouse cranes *43*
5.6 Number of rgv's on the terminal *45*
6 Computer simulation **47**
6.1 Problem analysis tool: PROSIM *47*
6.2 Definition of the components of the computer model *48*
6.2.1 The component Main 49
6.2.2 The component ship Generator 50
6.2.3 Import generator & Export generator 50
6.2.4 The component Berthmaster 51
6.2.5 The component Portainermaster 52
6.2.6 The component Rgvmaster 56
6.2.7 The component Trafficmaster 58
6.2.8 The class component Vessel 61
6.2.9 The class component Portainer 62

6.2.10 The class component Rgv ... 64
6.2.11 The class component Crane ... 66
6.3 The macros used in the program ... 67
6.4 The layout of the terminal as created in the Main ... 68
6.5 The traffic rules at the terminal ... 72
6.5.1 The priority rules ... 72
6.5.2 The driving of a rgv ... 73
7 Verification of the simulation program ... 77
7.1 Verification of algorithms ... 78
7.1.1 Verification of the allocation algorithms ... 78
7.1.2 Verification of route generating algorithms ... 81
7.2 Verification of the route of a travelling rgv ... 82
7.3 Interaction between different rgv's at crossing nodes ... 83
7.4 A qualitative analysis of the rgv cycle times ... 84
7.5 Portainer handling function ... 88
7.6 Influence on ship cycles when varying the number of rgv's ... 89
8 Simulation and results ... 91
8.1 Determination and description of the reference terminal ... 91
8.1.1 Standardisation of the container vessels ... 91
8.1.2 The standardisation of the terminal components ... 92
8.2 Performance indicators ... 93
8.3 Feeder vessel moored at Berth 1 ... 94
8.3.1 Unloading of a feeder vessel ... 95
8.3.2 Fast loading of a feeder vessel ... 97
8.3.3 Adjacent loading of a feeder ... 99
8.3.4 Shuffling from a feeder berth ... 101
8.3.5 Recapitulation of the feeder vessel simulation runs ... 103
8.4 Experiment: Mega vessel moored at Berth 2 ... 105
8.4.1 The unloading of the mega vessel ... 106
8.4.2 The fast loading of a mega vessel ... 108
8.4.3 Adjacent loading of a mega vessel ... 110
8.4.4 The shuffling from Berth 2 ... 112
8.4.5 Recapitulation of the mega vessel simulation runs ... 114
8.5 Simulation runs with 3 occupied berths ... 116
8.5.1 Experiment 1 ... 117
8.5.2 Experiment 2 ... 121
8.5.3 Experiment 3 ... 125
8.5.4 Experiment 4 ... 128
8.5.5 Recapitulation on the simulation runs with 3 occupied berths ... 130
8.6 Simulation runs of variations on the reference terminal ... 134
8.6.1 Increasing the number of adjacent loading rgv's to 12 ... 135
8.6.2 The loading and unloading of a vessel using different portainer cycle times ... 138
8.6.3 Stacking crane cycle times ... 143
9 Discussion of the Simulation Results ... 145
9.1 Introduction ... 145
9.2 Interpretation of the simulation results ... 146
9.2.1 Handling performance ... 146
9.2.2 Influence of variations to the reference terminal ... 149
9.2.3 Crane occupancy rates ... 150
9.2.4 Consequences for the terminal area ... 151
9.2.5 Integration of rgv waiting rows on the terminal ... 151
9.3 Queuing theory ... 154
9.3.1 Mega vessel berth ... 154
9.3.2 Feeder berths ... 156
9.3.3 Yearly throughput ... 157
10 Conclusions ... 159
11 Recommendations ... 161
12 References ... 163

Foreword

This report concerns the study on the design and simulation of a high capacity container terminal. It has been carried out as a Master of Science thesis for section Ports and Terminals of the faculty of Civil Engineering and Geosciences at the Delft University of Technology. The appendices of this report have been divided into two separate volumes.

We would like to thank prof. ir. H. Lichteringen and ir. Groenveld of the section Ports and Terminals and prof. ir. J.C. Rijsenbrij of the section Transport technology of the faculty of Mechanical Engineering and Marine Technology for their support and critical notes. We would like to thank prof. Hoong Chor Chin for the efforts he made as this study has been partly carried out at the National University of Singapore.

Hein van Hees
Thies Visscher
Delft, December 2000

Contents

Foreword

Summary

List of Figures

List of tables

List of Definitions

1 Introduction **1**
2 Problem analysis **5**
2.1 Objective *5*
2.2 Problem approach *5*
3 Assumptions and boundary conditions concerning the terminal **7**
3.1 Containers *7*
3.2 The vessels *8*
3.2.1 The Mega vessel 8
3.2.2 The feeders 9
3.3 The berths *9*
3.4 Handling strategies of the portainers *10*
3.5 Terminal strategies *11*
3.6 Arrival cycles of the vessels *15*
3.7 Required container storage capacity *16*
4 Choice of the terminal components and their capacities **17**
4.1 Portainers *17*
4.2 Container storage system *21*
4.2.1 Description of the chosen warehousing variant 22
4.2.2 Adaptations to the chosen warehousing concept 25
4.3 Horizontal transportation *28*
4.3.1 Rubber tired automated guided vehicles (agv's) 29
4.3.2 Rail guided vehicles (rgv's) 30
4.3.3 Criteria and choice 31
5 The design of the terminal Layout **33**
5.1 Positioning of the container warehouses *33*
5.2 Positioning and capacity of the rgv lanes *34*
5.2.1 Inter-berth lanes 34
5.2.2 Yard-lanes 36
5.3 Additional terminal design aspects *38*
5.4 Handling strategies applied to the terminal Layout *40*
5.4.1 The unloading of a container vessel 40
5.4.2 The adjacent loading of a mega vessel 41
5.4.3 The adjacent loading of a feeder vessel 42
5.4.4 The shuffle process on the terminal 42
5.5 Number of warehouse cranes *43*
5.6 Number of rgv's on the terminal *45*
6 Computer simulation **47**
6.1 Problem analysis tool: PROSIM *47*
6.2 Definition of the components of the computer model *48*
6.2.1 The component Main 49
6.2.2 The component ship Generator 50
6.2.3 Import generator & Export generator 50
6.2.4 The component Berthmaster 51
6.2.5 The component Portainermaster 52
6.2.6 The component Rgvmaster 56
6.2.7 The component Trafficmaster 58
6.2.8 The class component Vessel 61
6.2.9 The class component Portainer 62

6.2.10 The class component Rgv ... 64
6.2.11 The class component Crane ... 66
6.3 The macros used in the program ... 67
6.4 The layout of the terminal as created in the Main ... 68
6.5 The traffic rules at the terminal ... 72
6.5.1 The priority rules ... 72
6.5.2 The driving of a rgv ... 73
7 Verification of the simulation program ... 77
7.1 Verification of algorithms ... 78
7.1.1 Verification of the allocation algorithms ... 78
7.1.2 Verification of route generating algorithms ... 81
7.2 Verification of the route of a travelling rgv ... 82
7.3 Interaction between different rgv's at crossing nodes ... 83
7.4 A qualitative analysis of the rgv cycle times ... 84
7.5 Portainer handling function ... 88
7.6 Influence on ship cycles when varying the number of rgv's ... 89
8 Simulation and results ... 91
8.1 Determination and description of the reference terminal ... 91
8.1.1 Standardisation of the container vessels ... 91
8.1.2 The standardisation of the terminal components ... 92
8.2 Performance indicators ... 93
8.3 Feeder vessel moored at Berth 1 ... 94
8.3.1 Unloading of a feeder vessel ... 95
8.3.2 Fast loading of a feeder vessel ... 97
8.3.3 Adjacent loading of a feeder ... 99
8.3.4 Shuffling from a feeder berth ... 101
8.3.5 Recapitulation of the feeder vessel simulation runs ... 103
8.4 Experiment: Mega vessel moored at Berth 2 ... 105
8.4.1 The unloading of the mega vessel ... 106
8.4.2 The fast loading of a mega vessel ... 108
8.4.3 Adjacent loading of a mega vessel ... 110
8.4.4 The shuffling from Berth 2 ... 112
8.4.5 Recapitulation of the mega vessel simulation runs ... 114
8.5 Simulation runs with 3 occupied berths ... 116
8.5.1 Experiment 1 ... 117
8.5.2 Experiment 2 ... 121
8.5.3 Experiment 3 ... 125
8.5.4 Experiment 4 ... 128
8.5.5 Recapitulation on the simulation runs with 3 occupied berths ... 130
8.6 Simulation runs of variations on the reference terminal ... 134
8.6.1 Increasing the number of adjacent loading rgv's to 12 ... 135
8.6.2 The loading and unloading of a vessel using different portainer cycle times ... 138
8.6.3 Stacking crane cycle times ... 143
9 Discussion of the Simulation Results ... 145
9.1 Introduction ... 145
9.2 Interpretation of the simulation results ... 146
9.2.1 Handling performance ... 146
9.2.2 Influence of variations to the reference terminal ... 149
9.2.3 Crane occupancy rates ... 150
9.2.4 Consequences for the terminal area ... 151
9.2.5 Integration of rgv waiting rows on the terminal ... 151
9.3 Queuing theory ... 154
9.3.1 Mega vessel berth ... 154
9.3.2 Feeder berths ... 156
9.3.3 Yearly throughput ... 157
10 Conclusions ... 159
11 Recommendations ... 161
12 References ... 163

Summary

The container traffic growth worldwide continues to increase. To keep up with the growth, shipping companies keep enlarging container ship size. The handling speed of the terminal will especially be a critical factor for the largest vessels. Terminals have to be able to serve these ships as best as possible. To keep the dwell time of the ships visiting a terminal to a minimum, the handling capacities on the terminal must get bigger. To achieve a maximum effectiveness of portainer productivity in relation to the land side of the terminal, the horizontal transportation and the stacking of the containers must be optimised. The amount of land available for expanding an existing terminal or creating a new terminal is scarce. When designing a terminal this is a very important factor to take into account. To minimise operating costs and maximise efficiency, automation of the terminal also is a factor to take into account.

The goal of this study has been to design a container transhipment terminal capable of handling mega container vessels and analysing the logistic processes that take place on this terminal. The analysis will be performed by a computer simulation in the simulation language PROSIM.

The designed transhipment terminal consists of three berths, two feeder berths and one mega vessel berth. The ship to shore container handling takes place by using portainers. The horizontal transportation takes place by rail mounted automated guided vehicles (rgv's) that use linear motor based transfer technology as their source of propulsion. The rgv's take corners by having their wheels rotated by four disks integrated in the rail system. Container storage takes place by stacking containers in racks placed in warehouses. This allows random access to the containers eliminating the time consuming shuffling.

With these components a terminal layout has been designed. Based on this layout a computer simulation has been carried out in order to analyse the logistic processes on the terminal.

Having assumed an average portainer capacity of 1 move per minute the achieved handling rates lie between 194 moves per hour and the 236 moves per hour for the feeder vessel berths, where the handling takes place by four portainers.
The handling rates on the mega vessel berth lie between the 300 moves per hour and the 352 moves per hour. Here the vessel handling takes place by 6 portainers. Portainer occupancy rates of 95-98% have been achieved during the unloading process of the container vessels.

List of Figures

Figure 1.1, development of container vessels 1
Figure 1.2, hub and spoke system 2
Figure 1.3, map of Singapore and its region South-east Asia 3
Figure 3.1, mega vessel 8
Figure 3.2, tangent berth [1], perpendicular berth [2] and a combination [3] 10
Figure 3.3, schematisation of transhipment terminal 11
Figure 3.4, a three berth terminal 12
Figure 3.5, the fast unloading strategy/adjacent loading strategy 13
Figure 3.6, adjacent unloading strategy and fast loading strategy 13
Figure 3.7, the fast loading, fast unloading and shuffle strategies 14
Figure 4.1, Dual hoist system with a fixed platform 17
Figure 4.2, dual hoist system with elevating platform 18
Figure 4.3, double trolley system 18
Figure 4.4, elevating girder crane 19
Figure 4.5, Paceco supertainer 19
Figure 4.6, the portainer cycle times 20
Figure 4.7, cross-section of a warehouse 22
Figure 4.8, top view of a warehouse 22
Figure 4.9, the storage rack of a warehouse 23
Figure 4.10, front view of the lifting process of a shuttle 24
Figure 4.11, the cradle of the elevating transfer vehicle 24
Figure 4.12, a cross section of two warehouses 26
Figure 4.13, top view of the warehouse with adaptations 26
Figure 4.14, agv turning radius 29
Figure 4.15, rgv taking a turn 30
Figure 5.1, first design phase 34
Figure 5.2, schematisation of inter-berth rgv flow when three vessels are being adjacent loaded. 35
Figure 5.3, Layout of Berth 2 including inter-berth lanes 35
Figure 5.4, Layout version 1 including warehouse lanes 36
Figure 5.5, layout version 2 including warehouse lanes 37
Figure 5.6, side view of the portainer 38
Figure 5.7, Berth 1 39
Figure 5.8, unloading strategy of a container vessel 40
Figure 5.9, adjacent loading from Yard 1 to Berth 2 41
Figure 5.10, adjacent loading of a feeder vessel from Yard 2 42
Figure 5.11, portainer production during unloading/fast-loading 43
Figure 5.12, Portainer production during unloading fast loading on Berth 1 and adjacent loading on Berth 2 44
Figure 5.13, portainer production during unloading/fast-loading on berth 2 and adjacent loading on berth 1 and berth 3 44
Figure 6.1, the process of Main 49
Figure 6.2, process of the ship Generator 50
Figure 6.3, Process of the Berthmaster 51
Figure 6.4, mega vessel moored at Berth 2 52
Figure 6.5, feeder vessel moord at Berth 1 52
Figure 6.6, a feeder moored at Berth 3 53
Figure 6.7, feeder vessels moored at Berth 1 & Berth 3 53
Figure 6.8, feeder vessel moored at Berth 1 and a mega vessel moored at Berth 2. 54
Figure 6.9, feeder vessel moored at Berth 3 and a mega vessel moored at Berth 2 54
Figure 6.10, Three vessels moored 54
Figure 6.11, process of the Rgvmaster 57
Figure 6.12, process of the Trafficmaster 58
Figure 6.13, possibilities when a rgv is working for a vessel 59
Figure 6.14, process of the class component vessel 61

Figure 6.15, process of the class component portainer 62
Figure 6.16, flow chart of the portainer while it is going to the right position at the quay 63
Figure 6.17, process of the rgv 65
Figure 6.18, process of the class component crane 66
Figure 6.19, the creation of a main track by the Main 68
Figure 6.20, input file for creating warehouse lanes 69
Figure 6.21, the creation of a stack 69
Figure 6.22, input file for creating shortcuts 70
Figure 6.23, the creation of a shortcut 70
Figure 6.24, input file for creating a platform 71
Figure 6.25, the creation of 2 platforms and a portainer location 71
Figure 6.26, the claiming of nodes when driving on the main track 72
Figure 6.27, space for waiting queue at node 44 73
Figure 6.28, flow chart of the driving process of a rgv 74
Figure 6.29, the driving of the rgv 75
Figure 7.1, allocation of stacks in Yard 2 during the unloading of a mega vessel 79
Figure 7.2, allocation of the stacks during the adjacent loading of a feeder vessel moored at Berth 1 80
Figure 7.3, rgv entries in the stacks during a shuffle process 81
Figure 7.4, the queue behind a claimed node 83
Figure 7.5, unloading cycle of a rgv 84
Figure 7.6, rgv unloading cycle times when unloading a vessel 85
Figure 7.7, loading cycle of an rgv 85
Figure 7.8, rgv cycle times when adjacent loading a vessel 86
Figure 7.9, shuffling cycle of a rgv 87
Figure 7.10, rgv cycle times during shuffling 87
Figure 7.11, the portainer cycle times 88
Figure 7.12 portainer productivity during the unloading of a feeder vessel 89
Figure 7.13, portainer productivity when adjacent loading using a different number of rgv's 90
Figure 7.14, Portainer 1 queue using 15 rgv's per portainer during the adjacent loading of a feeder 90
Figure 8.1, a feeder vessel moored at Berth 1 94
Figure 8.2, the unloading of a feeder vessel 95
Figure 8.3, The portainer occupancy rates during the unloading of a feeder vessel at Berth 1 95
Figure 8.4, rgv queue under the portainers at Berth 1 during the unloading of a feeder 96
Figure 8.5, rgv waiting queues on warehouse lane 22 96
Figure 8.6, the fast loading of a feeder vessel 97
Figure 8.7, the portainer occupancy rates during the fast loading of a feeder vessel 97
Figure 8.8, rgv waiting rows under portainers at Berth 1 when loading fast 98
Figure 8.9, the adjacent loading of a Feeder 99
Figure 8.10, the portainer occupancy rates when loading a feeder from an adjacent berth ... 99
Figure 8.11, the rgv waiting queues under the portainers at Berth 1 when adjacent loading 100
Figure 8.12, the shuffling from Yard 1 101
Figure 8.13, rgv queue at node 290 102
Figure 8.14, rgv waiting row on warehouse lane 21 102
Figure 8.15, mega vessel moored at Berth 2 105
Figure 8.16, the unloading of a mega vessel 106
Figure 8.17, the occupancy rate of portainers 5 to 10 106
Figure 8.18, rgv waiting queues under the portainers on Berth 2 107
Figure 8.19, the fast loading of a mega vessel 108
Figure 8.20, Portainer occupancy rates during the fast loading of a mega vessel. 108
Figure 8.21, the rgv waiting queues under the portainers when loading a mega vessel 109
Figure 8.22, the adjacent loading of a mega vessel 110
Figure 8.23, Portainer occupancy rate when loading a mega vessel from adjacent berths .. 110
Figure 8.24, the rgv waiting queues under the portainers during the loading from an adjacent berth 111
Figure 8.25, shuffling from Berth 2 112
Figure 8.26, the rgv queue at node 363 when shuffling from Berth 2 112
Figure 8.27, the rgv queue at node 393 during shuffling from Berth 2 113

List of Tables

Table 3.1, Results of cycle times calculations 15
Table 4.1, operational properties of a rgv 29
Table 5.1, Warehouse crane occupancy rates 44
Table 5.2, number of needed rgv's per portainer 45
Table 8.1, the undisturbed container productions for Berth 1 103
Table 8.2, warehouse crane occupation rates during the undisturbed handling of a feeder vessel 104
Table 8.3, berth productions for Berth 2 114
Table 8.4, warehouse crane occupation rates during the undisturbed handling of a mega vessel 115
Table 8.5, Experiments with 3 occupied berths 116
Table 8.6, berth productions during adjacent loading on all berths 118
Table 8.7, berth productions 122
Table 8.8, berth productions 126
Table 8.9, handling rates during simulation on 3 berths 130
Table 8.10, comparison between the undisturbed and maximum service times 130
Table 8.11, warehouse crane occupancy rates 131
Table 8.12, description of alterations to the reference terminal 134
Table 8.13, handling productions with 12 rgv's 135
Table 8.14, the mean portainer cycle times 139
Table 8.15, portainer productions during the basic run and the crane experiment 143
Table 9.1, the handling rates and service times 146
Table 9.2, comparison between undisturbed situation and the ideal preliminary calculation 147
Table 9.3, comparison between undisturbed situation and the ideal preliminary calculation 147
Table 9.4, comparison between disturbed situation and the ideal preliminary calculation 148
Table 9.5, comparison between disturbed situation and the ideal preliminary calculation 148
Table 9.6, portainer productions of the basic run and the crane experiment 150
Table 9.7, number of cranes and their occupancy rates 150
Table 9.8, simulation results for the waiting rows and applied waiting row length 151

List of Definitions

Berth:	*Location where the vessels will moor.*
Yard:	*Behind each berth there is a yard containing the container storage.*
Stack:	*The yards are divided into two stacks; A stack that holds the 20' containers and a stack that holds the 40' containers.*
Portainer lane:	*Rgv are handled by a portainer on the portainer lanes located behind the portainers.*
Yard lane:	*A yard lane is a perpendicular to the berth orientated lane that allows the rgv's to enter the yard.*
Inter-berth lane:	*An inter-berth lane provides the connection between the 3 yard-berth combinations.*
Warehouse lane:	*A warehouse lane allows the rgv's to enter the container warehouses from a yard lane.*
Reference terminal:	*The terminal defined as standard terminal. The reference terminal is the basis of each simulation run.*

1 Introduction

The transportation of goods in containers has revolutionised the maritime and port business across the world. Since the introduction of container transportation in the sixties, the development of container transport has increased dramatically. Nowadays container transportation is a full-grown business with a high level of technological means.

This increase of container transportation and subsequently the growing capacity of container vessels over the last couple of years has forced terminal operators to constantly search for means to increase their handling capacity. Already 8,000 and 8,000+ TEU[1] mega vessels are on the drawing boards. Add to this the ever-growing number of 6,000 and 6,600 TEU ships already in operation and it becomes clear that shipping operators are increasing their ship sizes at a fast rate, as illustrated in Figure 1.1.

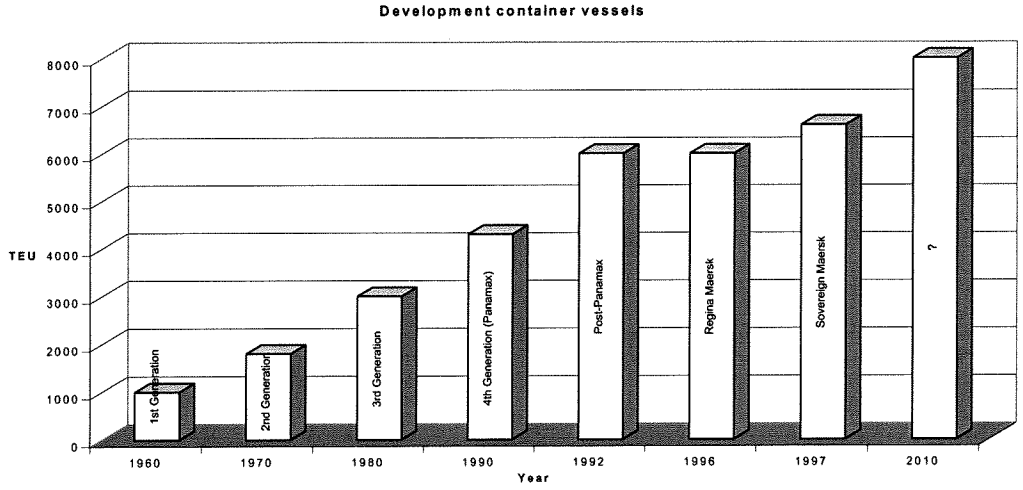


Figure 1.1, development of container vessels

The size of these ships and the enormous costs per hour, demand an optimal turn around time. Due to the increase in scale, terminals with high loading and unloading capacities will have the advantage in competition. Especially those terminal operators wishing to maintain their 'hub port' status should upgrade their facilities, as shipping companies will prefer to attend terminals where vessel handling times are short. The hub ports offering the fastest, most reliable service at lowest overall cost will attract the biggest share of container traffic.

Overpopulation and industrial development mean that the amount of land available for the construction or expansion of container terminals is minimal, whilst environmental concerns place new restrictions on development operations. Particularly in Singapore, Hong Kong and perhaps in lesser degree also Rotterdam port area is a scarce commodity. Massive land reclaiming projects are needed for port expansion. It is therefore essential that along with increased handling speed the land usage of the terminal needs to be minimised in the terminal design.

Besides the need for higher handling speed and optimisation of land usage a third trend in port development can be distinguished. In their attempts to control the complete range of terminal operations, ports see their efforts frustrated by the variability of man operated vehicles. Automation will eliminate this and should facilitate the development of efficient

[1] TEU = **T**wenty foot **E**quivalent **U**nit

scheduling of operating systems. Automation of terminal operations also reduces operating costs by removing manpower besides satisfying the growing demand for efficiency.

The expectation is that in a number of years, mega vessels with capacities of 8,000 TEU or more will maintain the intercontinental container transportation between ports with a hub status. Large amounts of containers will have to be handled at each mega vessel call. Feeder vessels responsible for the regional distribution of containers will serve the regional ports, as is schematised in Figure 1.2.

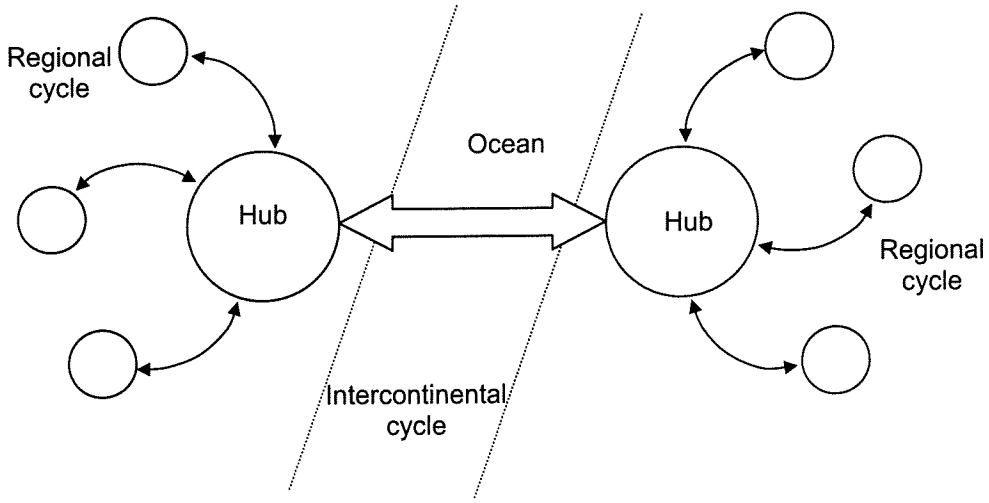


Figure 1.2, hub and spoke system

A typical hub port is the transhipment terminal. A transhipment terminal has no or limited hinterland demanding containers. The container flows to and from the terminal are limited to the unloading and the loading of the container vessels.
Good examples of transhipment hub ports are the container terminals of Singapore. Goods manufactured in Southeast Asia are brought to Singapore from regional ports by feeder vessels. From Singapore the loaded containers are distributed all over the world. In this study the situation as it occurs in Singapore will be used as a basis. Figure 1.3 demonstrates the central location of Singapore within Southeast Asia.

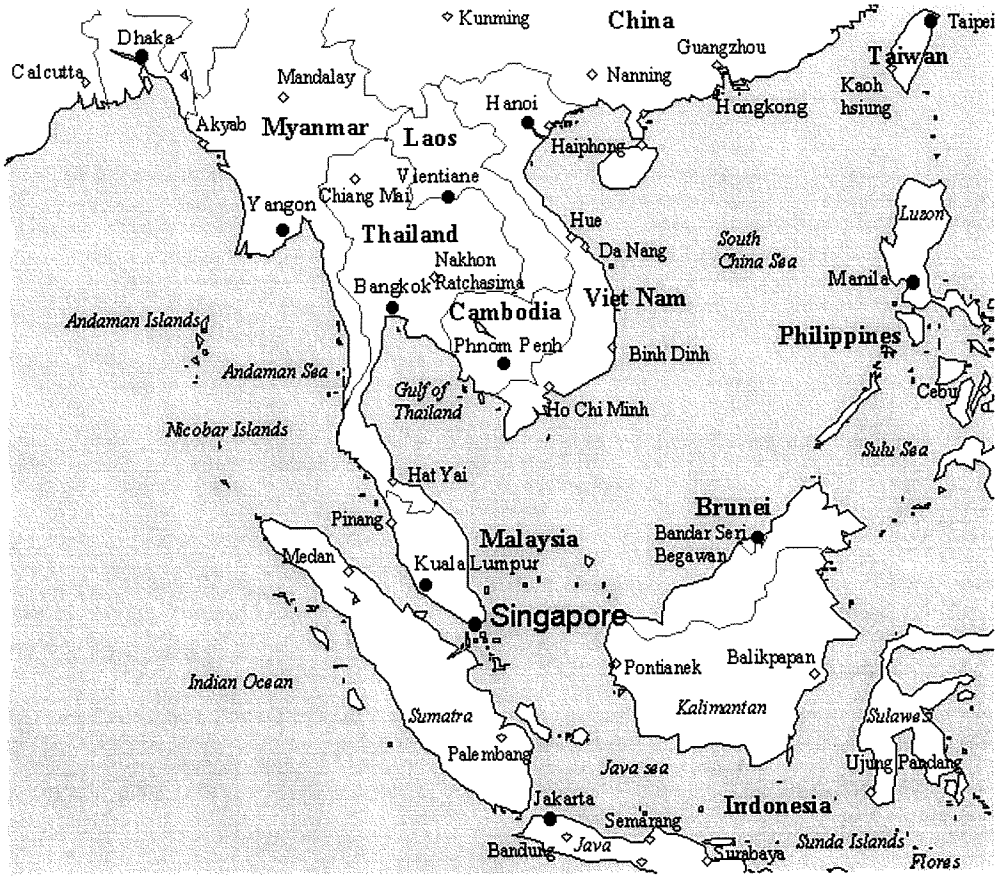


Figure 1.3, map of Singapore and its region South-east Asia

This study contains the following chapters. The problem analysis will be dealt with in Chapter 2, followed by a description of the specifications on and around the terminal in Chapter 3. Chapter 4 describes the components portainer, container storage system and horizontal transportation. The chosen components will be integrated into a terminal layout in Chapter 5. Chapter 6 describes the computer simulation model, which will be verified in Chapter 7. The standard reference terminal is defined in the first part of Chapter 8 and will be analysed by means of computer simulation in the second part of Chapter 8. Discussion of the results is carried out in Chapter 9, followed by the conclusions in Chapter 10 and the recommendations in Chapter 11.

2 Problem analysis

2.1 Objective

The objective of this study is to design and simulate a transhipment container terminal:

- Capable of handling mega container vessels of 8000 TEU.
- With a high level of automation.
- While maximising container throughput.
- While minimising the land usage.
- With the intention not to invent new kinds of terminal equipment. The aim is to use only existing technology.

The attention will be especially focussed on the container flows at the terminal and the achievements of the different components in relation to each other.

2.2 Problem approach

The container terminal will have comparable circumstances as to those in Singapore. This means that any transportation of containers to hinterland can be neglected and will not be taken into account, making it a pure transhipment terminal called at by regional feeders and intercontinental container vessels.

The first phase will be the design process of the transhipment container terminal by first choosing the terminal components followed by integrating these components into a terminal layout. After the design of the terminal is complete, the operational processes of the terminal will be modelled and simulated with a computer simulation program.

The second phase in the design of the terminal will be to simulate the processes of the terminal with the computer simulation program PROSIM. This way the terminal capacities can be analysed and quantified.

This study will be highly concentrated on the processes that take place in the terminal. This means that a high level of simulation accuracy will be achieved concerning the different components at the terminal. A result of this design approach is that in the simulation phase, the terminal will be primarily tested by temporarily increasing and decreasing the pressure on the terminal and not by performing simulation runs that comprises several weeks of arriving container vessels.

This makes it more difficult to make comments on vessel turn around times, stack capacities and yearly throughput, as longer runs are required. But it does give the possibility to look at the specific processes at the terminal in detail.

3 Assumptions and boundary conditions concerning the terminal

During the design process of a terminal some boundary conditions must be understood and assumptions must be made. These boundary conditions concern the type of vessels that call on the terminal and the different kinds of containers they deliver. The assumptions concern some choices with respect to the number of berths and the handling strategies at the terminal.

3.1 Containers

Currently a number of different common container types can be distinguished by size and sort of load. An inventory has been made by [L.1].

- Standard containers.
  Containers that belong to this group have a length of 20' or 40' and a width of 8'. The height varies between 8' and 9'6". Electrically cooled containers or reefers also belong to this group, and need special facilities to provide them with electricity. 98 % of all containers belong to this category. The amount of 20' containers handled on the terminal is the same as the amount of 40' containers handled on the terminal.
- Containers with a 45' length.
  Except for the length these containers do not differ from the standard containers. Of all handled containers a mere 1% belongs to this category.
- Off standard A containers.
  Containers belong to this class if the width exceeds the standard width with half a meter. If the length of a container exceeds a 40' standard container with one meter or if the length exceeds a 45' container by half a meter it also belongs to this category. If the standard height is exceeded by a meter, if the height exceeds 3.6 meters or if the container weighs more than 41.5 tons the container also belongs to this category. 0.8 % of all containers belongs this category.
- Off standard B containers.
  If the dimensions of the container do not correspond with any of the above classes, it automatically belongs to this category. 0.2 % of these containers belong to this group of off-standard containers.
- IMO containers.
  Containers containing hazardous loads are classified as IMO containers. Containers belonging to IMO class 1 and 7 require a special position on the container vessel. The dimensions of these containers however almost always correspond to standard containers. 0.1 % of the containers belong to this category.

Both the off standard B and the certain IMO classes of containers require off-standard transportation facilities. The off standard exceptions will not be included in the simulation program. The simulation will only include the standard 20' and 40' containers. 45' containers will be treated the same way as 40' containers.

3.2 The vessels

3.2.1 The Mega vessel

As demand for container transportation increased over the years the vessel size followed. Currently the feasibility of mega vessels with capacities varying from 8,000 TEU up to 10,000 TEU is being studied on, while 6,700 TEU vessels are already operational.
In this study we will derive the design vessel from the FAMAS project, being an 8.000 TEU mega container vessel. The mega vessel as used in FAMAS is based on a number of assumptions stated in [L.2], [L.3] and [L.4].

- The length of the mega vessels will not exceed 350 meters due to shipping characteristics.
- The depth of the mega vessels will be restricted to 14 meters considering the water depth of most harbours.
- Inside the hold of the mega vessels there is room for stacking 9 containers in height.
- On the decks the maximum stacking height is 7 containers.
- There are six bays behind the wheelhouse allowing two portainers to handle the rear of the vessel.

Based on these assumptions the following vessel properties have been found.

Capacity:	8,000 TEU
Length:	335 m.
Width:	53 m.
Draught:	14 m.

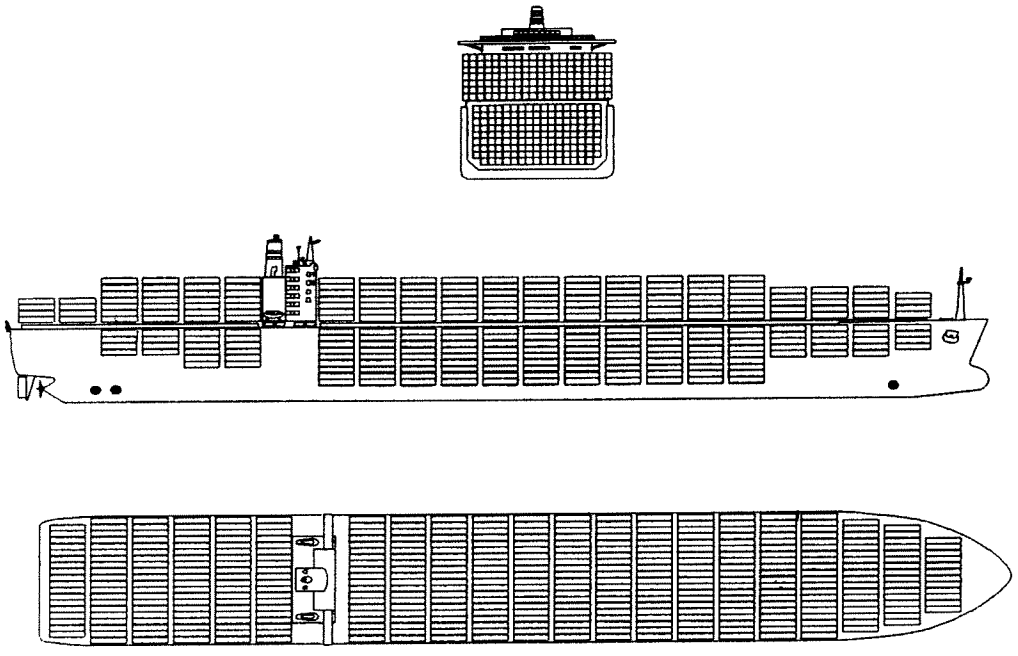


Figure 3.1, mega vessel

These vessels will be responsible for the intercontinental hub to hub transportation of the containers.

3.2.2 The feeders

The feeders take care of the regional distribution of containers and attend regional ports. Some assumptions concerning the feeders are made.

- The feeders are 2nd or 3rd generation container vessels. These vessels can carry around 1600 TEU
- The length of the feeders varies between 201 and 230 meters.
- The depth of the feeders will not exceed 11.5 meters
- The width of the feeders will not exceed 32 meters.

This study has been based on the situation as it exists in Singapore, so the share of containers of the terminal on the mega vessels will correspond to the current share of containers on the intercontinental container trade. This share is roughly 60 percent resulting in 4800 TEU to be unloaded and 4800 TEU to be re-loaded.

A feeder vessel is supposed to only serve this transhipment terminal and will therefore unload its entire cargo at the terminal. The feeder will then be completely loaded again with empty containers.

3.3 The berths

The number of berths

Generally a container terminal and especially hub terminals have multiple berths. By combining the portainers, container storage and horizontal transportation, the larger terminals will be more efficient and flexible. To include these synergy advantages into the initial design and the computer simulation, the choice has been made to design a multiple berth terminal. Increasing the number of berths consequently will also increase the complexity and the magnitude of the simulation program. A compromise between the complexity of the simulation program and the inclusion of the synergy advances has been found by designing a three-berth terminal.

The orientation of the berths

Orientation of berths can roughly be divided into the conventional tangent orientation and perpendicular orientation or slip berth. Lately a lot of research has been done to the nautical, hydrodynamic and logistic aspects of slip berths. A big advantage is that a vessel can be handled from both sides. Unfortunately, docking a ship between two quays in tidal water will not be an easy matter to arrange. Other disadvantages are:

- The berth occupancy and portainer occupancy is much lower as portainers cannot be transferred from one berth to another.
- The berth is not suitable for all vessel sizes.
- Twice as much quay walls are required.

As it is still uncertain if the advantages outweigh the disadvantages of the berths, the design and simulation of the terminal will be carried out with conventional tangent orientated berths as can be seen in Figure 3.2-1.

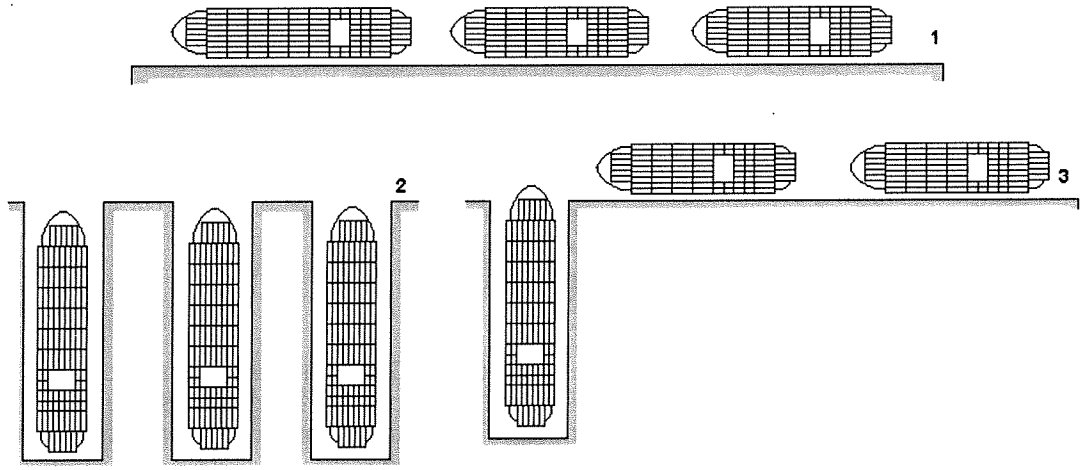


Figure 3.2, tangent berth [1], perpendicular berth [2] and a combination [3]

The berth length

Normally, the berth length depends on the length of the design vessel, which can be determined by extensive research on the arrival frequencies and the length of the visiting container vessels. As the type of the container vessels in this study are schematised and restricted, and arrival times will be left out of the computer simulation, a realistic and practical design vessel cannot be determined this way.

In practise only one mega vessel can be berthed at a time. The transhipment character of the terminal justifies to assume that the other two available berths are taken by feeders.

The complete berth length thus equals the length of a mega vessel plus the length of two feeder vessels multiplied with a margin of 10 percent needed for mooring. This amounts to a total length of 875 meters for the 3 berths.

3.4 Handling strategies of the portainers

The unloading and loading of a container vessel is determined by the handling plan. Different strategies can be applied on a handling plan. These strategies are divided in bay level strategies and ship level strategies.

Bay level strategies.

When working at bay level there are a few strategies to be distinguished. These strategies often depend on the shape and type of a container vessel and the distribution of the containers on the vessel.

- Working vertically in the vessel.
  This is only possible with a hatchcoverless vessel and when unloading the hold of a conventional vessel.
- Working in the width of the vessel, the grazing of the ship.
  The containers are unloaded and loaded layer by layer from one side of the ship to the other side of the ship.

Other than the above bay level strategies exist, but are difficult to carry out. The chosen bay level strategy used to handle the containers on the vessels in this study is the grazing of the ship.

Strategies at ship level

Three possibilities can be distinguished for the handling of the containers at ship level.

- Separate loading and unloading.
  A portainer starts unloading a bay. If the bay is unloaded the portainer will move to a next bay and resumes the unloading process. All the portainers in this process move in the same direction. If one portainer is finished unloading its hold the loading of the holds starts. The aim for all the portainers is to start the loading process simultaneously. The portainers now move in the opposite direction.
- Integrated loading and unloading.
  After a portainer has finished unloading a bay it starts loading it. In this case the portainer does not have to move as much, thereby increasing productivity. The flows of the combined import and export containers make the planning in the stack more difficult.
- Simultaneous loading and unloading.
  This method concentrates on the utilisation of the empty portainer spreader as it returns to the container vessel during the unloading procedure. By simultaneously unloading and loading a container vessel in this manner, the occupancy rates of the portainer spreader will increase. Accurate terminal planning is an extremely important condition for the correct implementation of this method.

In the simulation model the aim is to work according to the separate loading and unloading strategy combined with the grazing method.

3.5 Terminal strategies

The container flow through a terminal is a complex series of interconnecting activities. The output of the whole system depends on the capacity of the weakest link in that chain of activities. Basically a transhipment container terminal consists of three terminal components. The first component is the portainer, which is responsible for the loading and unloading of the vessels. The second component is the horizontal transportation, which takes the containers from the quay to the yard. The horizontal transportation directly influences the unloading and loading of the vessel. The container storage is the third component in the system.

In order to avoid congestion of the horizontal transportation units the production of the stack cranes must have a production comparable to the portainers. As this study involves the design of a new terminal, the container storage system can be favourably placed directly behind the stacks. Figure 3.3 gives a schematic and simplified representation of a container terminal, which is the basis of the design and simulation study.

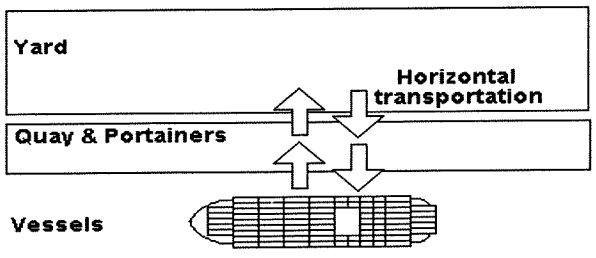


Figure 3.3, schematisation of transhipment terminal

As this study is based on the design of a transhipment terminal, containers can only enter or leave the terminal by vessel.

Berthing strategy

As stated before the terminal will be visited by two kinds of vessels, a regional feeder vessel and an intercontinental mega vessel. The three available berths have been divided into two feeder berths and one mega vessel berth.

The most efficient configuration of the berths is to reserve the middle berth, Berth 2, for the mega vessels. This way the relative travelling distances for the horizontal transportation units will be minimised.

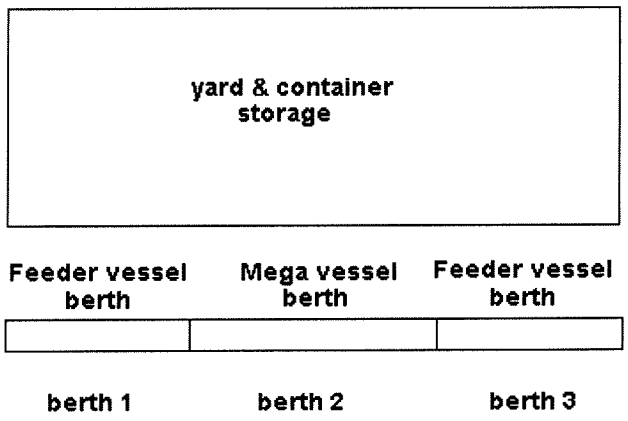


Figure 3.4, a three berth terminal

Container handling strategies

An important aspect of terminal design is determining the most efficient way of transporting the containers to and from their right locations in the stacks during the unloading and the loading of the vessel. A terminal with three or more berths can achieve considerable advantages by determining a favourable container handling strategy on the terminal. Especially when the stacks can be placed directly behind the berths as is the case in this study.

Considering the established berthing strategy, two relevant handling strategies will be discussed resulting in the container handling strategy that will be applied in this study. A consequence of the hub transhipment assumption and the berthing strategy is that feeder vessels mooring at Berth 1 or Berth 3 bring in all the regional containers to the terminal. These containers will stay in the stacks until they are eventually loaded onto a mega vessel moored at Berth 2. The opposite also applies. Containers brought in by mega vessels will eventually be loaded onto feeder vessels moored at Berth 1 or Berth 3. This naturally has consequences for the container handling strategies on the terminal.

Basically two different vessel unloading strategies can be applied, each resulting in a typical vessel loading strategy.

Fast unloading /adjacent loading strategy
If the unloaded containers are stacked directly behind the unloading berth, the travelling distances for the horizontal transportation units will be limited. Another advantage is that the encounters between horizontal transportation units operating on other berths and stacks are minimised. However, if for example a feeder vessel moored at Berth 1 unloads all its containers in Yard 1, the loading procedure of these containers onto a mega vessel moored at Berth 2 implies horizontal transportation from Yard 1 to Berth 2. This is called adjacent loading.

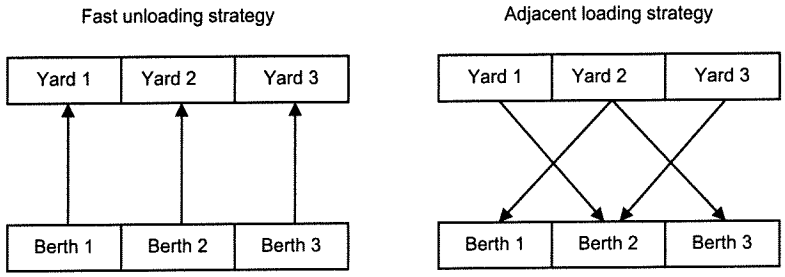


Figure 3.5, the fast unloading strategy/adjacent loading strategy

Adjacent unloading /fast loading strategy
Obviously the strategies can also be inversed. Applying adjacent unloading on the three berths will position the containers in the favourable stacks and facilitate the loading procedure.

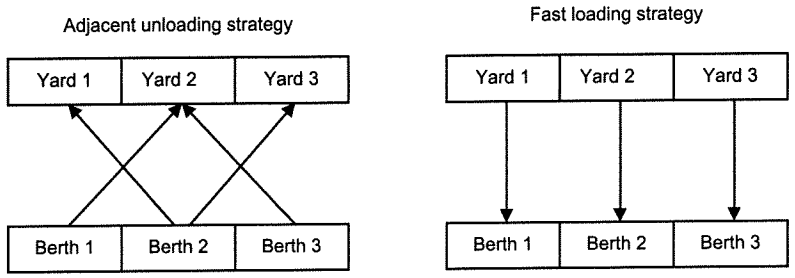


Figure 3.6, adjacent unloading strategy and fast loading strategy

Including the shuffle procedure to the container handling cycles.
Another container handling strategy is the fast unloading and the fast loading combination. This can only be achieved if the containers have been repositioned from the yard behind the unloading berth to the yard behind the loading berth during a separate procedure called shuffling.

By including a shuffle procedure to the container handling cycles, the containers can be unloaded and loaded very efficiently. From vessel service times point of view this is the most efficient strategy. But it is only possible if there is time available for shuffling. It also demands extra storage capacity in the stacks.

When the shuffle procedure is included, the containers that have been unloaded from the feeder vessels, moored at the berths in front of Yard 1 and Yard 3, will be shuffled to Yard 2 after the feeder vessel has left, ready to be fast loaded onto a mega vessel. Containers

unloaded from mega vessels moored at Berth 2 into Yard 2 will be shuffled to either Yard 1 or Yard 3 ready to be fast loaded onto the feeder vessels.

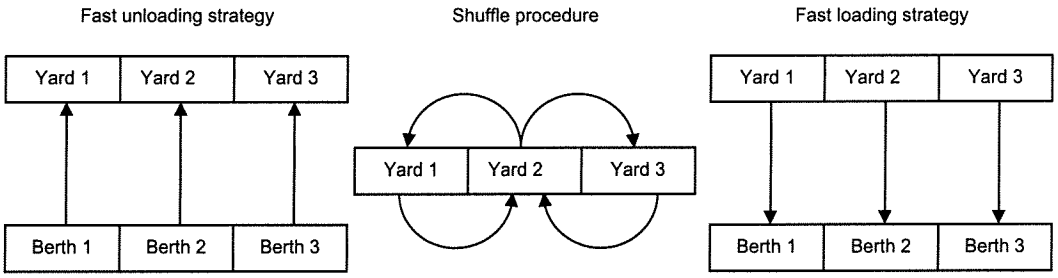


Figure 3.7, the fast loading, fast unloading and shuffle strategies

3.6 *Arrival cycles of the vessels*

Although this study will concentrate on the processes taking place at the terminal a virtual vessel arrival cycle has been assumed in order to make calculations concerning the required stack capacity.

In order to have a consistent simulation analysis the vessel types included in the simulation program have been limited to the two types described above. Of course, in reality a variety of vessels will be able to attend the terminal. The arrival cycles of the two types of vessels included in the simulation are partially determined by the demands dealt with above and the following boundary conditions.

- The occupancy rate of the terminal and the berths may not exceed 60 %, and should preferably be around 50 % in order to allow variations in arrival times without creating large congestion.
- The time needed to unload and load the vessels may not exceed 24 hours. This is including time needed to moor the vessel.
- The TEU-factor is estimated to be 1.5. This implies that the amount of 20' containers that is handled on the terminal is the same as the handled amount of 40' containers.
- There are separate berths for feeder vessels and for intercontinental vessels. In this study this has been assumed to be one berth for the mega vessels and two berths for the feeder vessels.

The pre-simulation calculation of the arrival cycles of the vessels can be seen in Appendix 3. The main results are given below.

Vessel type	Vessel capacity	Parcel size	Loading & unloading	Arrival cycles	Hours available per vessel	Portainers per vessel	Moves per portainer per hour
Mega vessels	8000 TEU	4725 TEU	9450 TEU	2 calls in 3 days	17.5	6	60
Feeder vessels	1600 TEU	1600 TEU	3200 TEU	2 calls in 1 day	8.75	4	60

Table 3.1, Results of cycle times calculations

The average required capacity of the portainers working on feeder and mega vessels is assumed 60 moves per hour as will be justified in the next chapter. The result of this calculation is that the berth occupancy rate of the mega vessel berth, Berth 2, is around the 54 %, while the occupancy rate of the feeder vessel berths, Berth 1 & 3 is around 45 %.

As follows from this vessel arrival scheme, the annual container throughput of the terminal will be around 4.6 million TEU. Per berth the annual throughput will amount to over 1.5 million TEU. If this amount is to be approached during simulation, the horizontal transportation and the container storage system must be able to process the production of the portainers. This configuration of ship arrival cycles can be seen as a benchmark for the terminal and will be checked at the end of this report.

3.7 Required container storage capacity

The method used to calculate the storage capacity needed in the stack takes the arriving cycles of the container vessels into account. Calling intervals of vessels and quantity of containers handled per vessel call are now introduced in the calculation. To calculate the stack capacity for transhipment terminal in this manner Watanabe [L.12] introduced the following formula:

$$C_0 = q_u + q_u(D_u - p)/D_u + q_u(D_u - 2p)/D_u + ... + q_u(D_u - np)/D_u + q_l + q_l(D_l - (p-1))/D_l + q_l(D_l - (2p-1))/D_l + ... + q_l(D_l - (np-1))/D_l$$

With: q_u = number of TEU to be unloaded from a mega vessel
q_l = number of TEU loaded onto a mega vessel
D_u = number of days that the feeder vessels are allowed to deliver containers in advance for the mega vessel
D_l = number of days that the feeder vessels are allowed to retrieve containers after the arrival of the mega vessel
p = periodic interval of mega vessels in days
n = 1, 2, 3, ...

Assumptions that have been made

- By simplifying the model to two kinds of ships, the 8,000 TEU mega container vessel and the feeder vessels, an initial storage calculation can be made.
- Assuming that the berth capacity of the terminal should be maximised within the boundaries of a 50 % berth occupancy rate, the arrival scheme of the 8,000 TEU vessels is once every 1.5 days. Feeder vessels arrive twice a day.
- The feeders are allowed to deliver containers starting from 5 days in advance.
- The share of the terminal on the mega vessels is assumed to be 60 %, resulting in 4,800 TEU to be unloaded of and the same number to be loaded again.
- Feeder vessels provide these containers resulting in 1,600 TEU to be unloaded and the same amount to be re-loaded per call.

Calculations

When the feeders arrive once every 1.5 days, and the deliveries for a mega container vessel start 5 days in advance the required stack capacity can be calculated.

Using the Watanabe formula the required capacity is: <u>24,000 TEU</u>

This calculation has been carried out in Appendix 4.

4 Choice of the terminal components and their capacities

The traditional terminal activities can be split up into three main components. The first component is the portainer that is responsible for unloading and loading the containers from and onto the vessels. The second component, the horizontal transportation, receives containers from the portainers during the unloading of a vessel and provides the portainers with containers during the loading of a vessel. Containers are stored in stacks. This can be seen as the third terminal component.

In the following chapters each of these terminal components will be discussed. The portainers will be analysed by introducing the portainer handling capacities. In this study the containers will not be conventionally stacked upon each other but will be stored in racks. Following the discussion, the most promising warehousing concept will be explained and improved for the specifications of this terminal. For the horizontal transportation component two alternatives will be compared using the two most important criteria in this study: land usage and production capacity.

4.1 Portainers

Along with the development of mega container vessels, crane designers have started to develop super productive portainers. The growth of production capacity of the conventional single hoist portainers by increasing trolley speed and acceleration resulting in a heavier framework seems to have reached its limits.

In the last couple of years new concepts have been introduced that should be able to increase crane productivity. Portainer productivity however must be seen as an integral part of the terminal system. After all, super productive portainers require super productive terminals.

Some of the concepts are:

Dual hoist system, fixed platform [L.5] & [L.9].
Splitting the trolley cycle into two processes and introducing a buffer on the fixed platform can significantly increase portainer productivity.

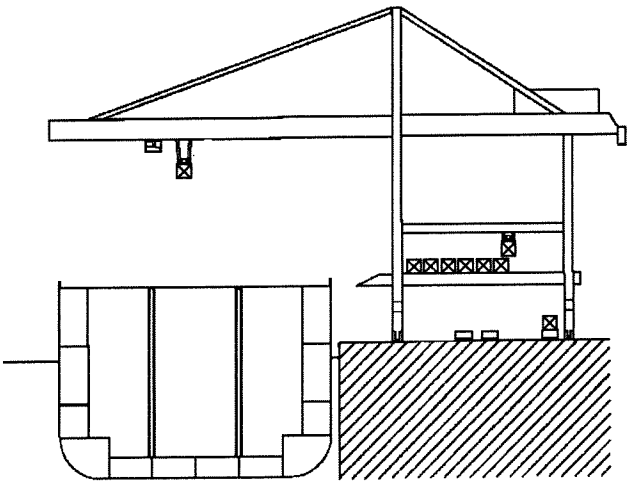


Figure 4.1, Dual hoist system with a fixed platform

Dual hoist system, elevating platform [L.8].
Lowering and elevating the platform of the portainer will decrease the travelling distance of the spreader thereby increasing portainer production. An independent simulation study described in [L.8] shows an average container production of 54 container moves per hour.

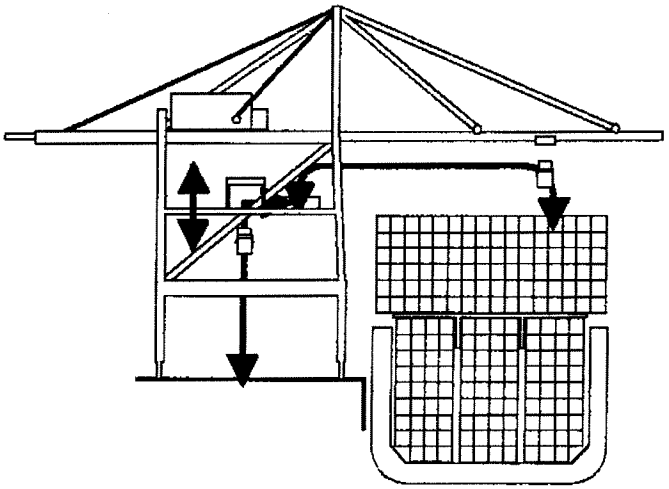


Figure 4.2, dual hoist system with elevating platform

Double trolley system [L.6].
Handling a container vessel with two trolleys operating independent of each other promises considerable handling productions. Average productions varying from 55 – 82.5 container moves per hour according to rough calculations.

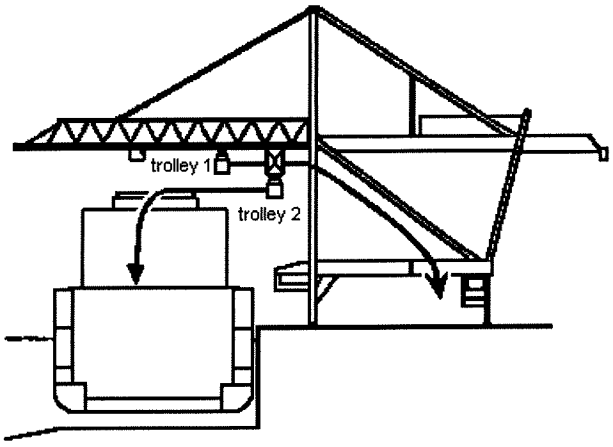


Figure 4.3, double trolley system

Twin-lift equipment / spreader [L.7].
By introducing twin-lift equipment the amount of TEU per move will increase from an average of 1.5 TEU/move to 2 TEU/move.

Elevating girder cranes [L.8].
By allowing the girder of a portainer to elevate and to lower along with the height of the stacked containers on a vessel the hoisting height is decreased, improving handling production.
[L.4] shows an average production varying from 55 container moves per hour to 66 container moves per hour.

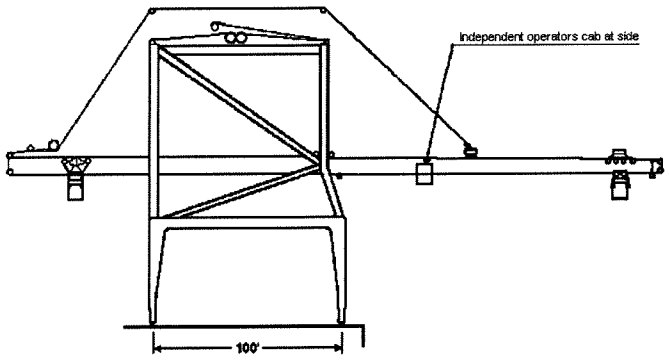


Figure 4.4, elevating girder crane

Ship trolley/shuttle/shore trolley system (Paceco[2]).
Paceco [L.9] claims to have the most productive concept with a maximum average production of up to 90 moves per hour on a hatchless Panamax ship. An independent study by [L.4] claims an average production of 67 moves per hour.

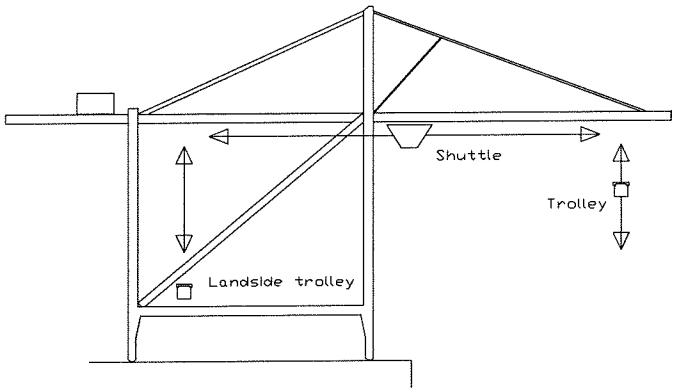


Figure 4.5, Paceco supertainer

The supertainer consists of three independent operating container carriage devices. During unloading the seaside trolley retrieves a container from a bay in the vessel and delivers it to the shuttle. The seaside trolley only makes a vertical movement while the shuttles carry out the horizontal movements. The landside trolley is responsible for the vertical movement necessary to deposit the container onto the horizontal transportation units.

The development of portainers in the last couple of years has shown a considerable increase in production. In this study no particular concept has been chosen, only an assumption on portainer productivity has been made. How the portainer productivity is modelled in the computer simulation will be explained in the next chapter.

[2] Paceco cooperation, the inventor of the Supertainer™

Determination of the portainer cycle times

In this study portainer cycle times from FAMAS [L.12] have been used. They are derived from a logistic analysis using a hatchcover vessel and a portainer called "jumbo container crane". The course of the portainer cycle times in time is given in Figure 4.6.

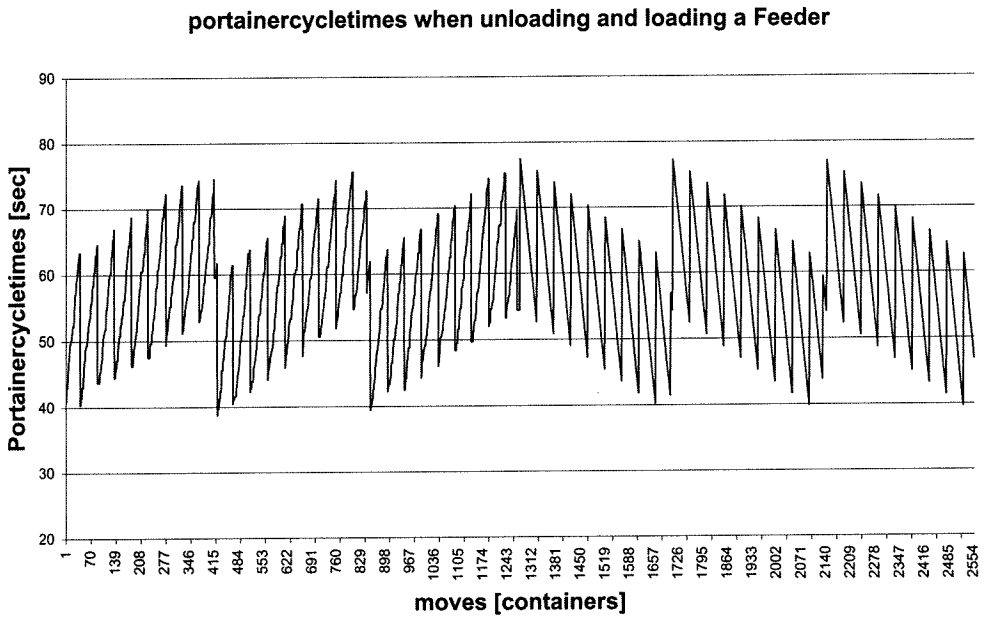


Figure 4.6, the portainer cycle times

These are the portainer cycle times when 4 portainers are unloading and subsequently loading a feeder vessel. Moves 1 to 1266 represent the unloading of the feeder vessel implying 1266 containers will have to be unloaded. Operating with 4 portainers means each portainer will unload approximately 315 containers.

The unloading starts with the container located on the position closest to the landside of the vessel. The time needed to retrieve this container is relatively short, as can be seen in Figure 4.6. After the first container has been unloaded, the portainer works horizontally towards the seaside of the vessel, unloading the top layer of containers. After the first layer has been unloaded the portainer takes away the second layer. This unloading process of one layer creates a tooth like shape in the portainer cycle times. In total there are 9 layers in this feeder vessel, which amounts to 9 teeth in the function. The cycletimes of the successive layers will become higher as the portainer reaches deeper inside the vessel.

After the first hold has completely been unloaded the portainer will transfer to the next hold and start all over again. Each portainer has 3 holds to handle as can be seen in Figure 4.6.

Moves 1267 to the end represent the loading of the vessel. During the loading of the feeder vessel the portainers will first deposit a container on the bottom of a hold at the seaside of the feeder vessel and work their way back to the landside of the vessel.

The average cycle time for the unloading and loading is 58 seconds, which could lead to a handling production of 62 moves per portainer per hour.

4.2 Container storage system

As land usage needs to be minimised, stacking height is an important design factor. Normally containers are stacked on top of each other. This conventional stacking method is restricted to nine containers in height by I ISO standards. Apart from the stacking height restrictions, there is another disadvantage to conventional stacking. When containers are stacked on top of each other shuffling and reshuffling of containers is a time consuming process, and takes more time with increasing stacking height.

Placing containers in "racks" or "warehousing" containers is a new approach in container storage. The big advantages are that stacking heights are not restricted to ISO limits, and even more important it reduces the time consuming shuffling. This random container access greatly decreases the retrieval and depositing time of a container in a stack. This subsequently permits quicker loading and unloading of container vessels. Another advantage is the reduction of labour cost associated with the handling of containers in the stack, as the system is completely automated. A disadvantage could be the high investment cost to be made for the expensive construction and foundation.

Momentarily several variants of container warehousing are being researched. For this study the most relevant and credible one [L.14] will be described in the following chapter. Using this container-warehousing variant as a basis, some alternations have been introduced in order to improve the production capacity.

4.2.1 Description of the chosen warehousing variant

The container warehouse as described in [L.14] consists of the following elements:
- overhead portal cranes
- storage racks
- stacking cranes called "elevating transfer vehicles" in [L.14]

Below, these different components of the warehouse will be discussed individually and in relation with each other. The bold numbers between the brackets used in the text refer to numbers used in the different figures.

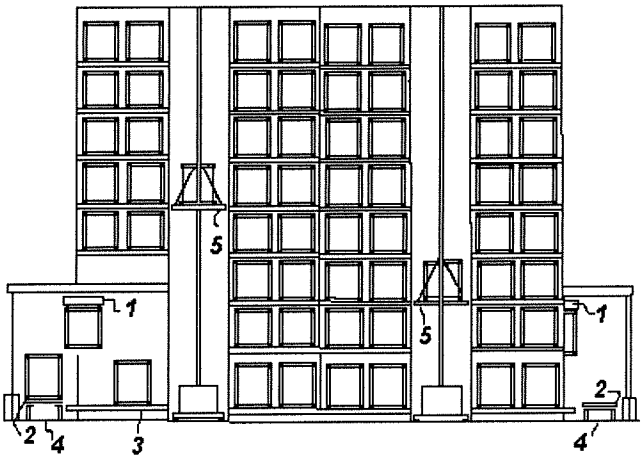


Figure 4.7, cross-section of a warehouse

Unloading/loading area
The overhead portal cranes **[1]** are responsible for the loading and unloading of the horizontal transportation units **[2]** as shown in Figure 4.7. The overhead portal crane consists of a rail-guided carriage supporting the hoist frame that holds a spreader. The spreader has twist locks, which engage the four fittings located on the top of the container. The portal crane is movable between positions above the portal buffer area **[3]** and the loading/unloading zone **[4]** of the horizontal transportation vehicle **[2]**.

As can be seen in Figure 4.8 these portal cranes **[1]** are distributed alongside of the warehouses. The horizontal transportation units **[2]** can also be distinguished. Container length **[6]** has a great influence on the warehouses and on the elevating transfer vehicle **[5]**.

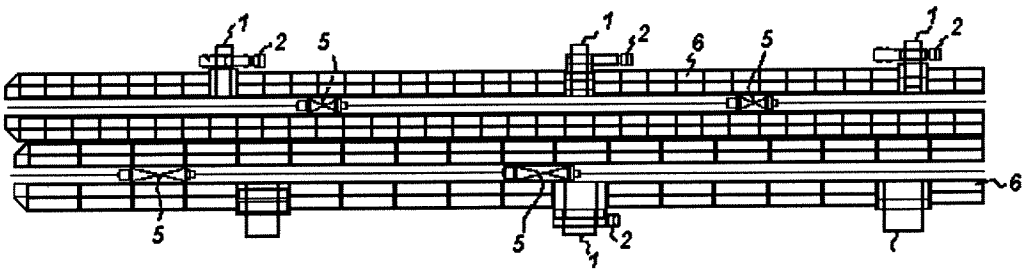


Figure 4.8, top view of a warehouse

Storage rack
The storage rack as shown in Figure 4.9 is the framework that holds the container **[7]**. This rack consists of a system of vertically and horizontally orientated metal beams **[8]** forming several container storage bays. Each bay is provided with supporting brackets **[9]** for supporting the container at its four lower corners. The bays have a depth sufficiently large to store two containers. This could make occasional shuffling necessary.

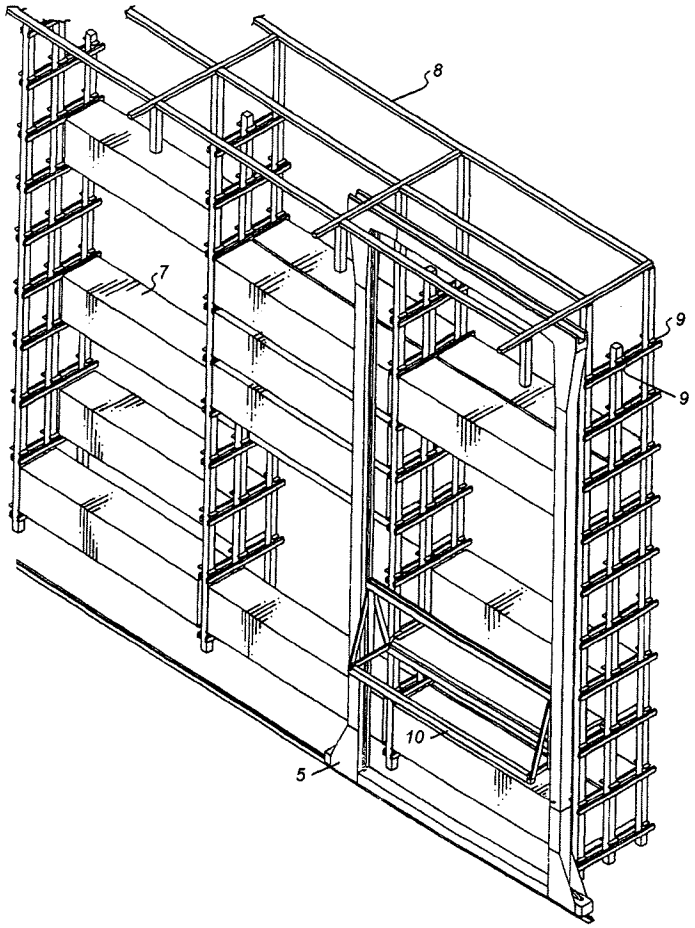


Figure 4.9, the storage rack of a warehouse

Elevating transfer vehicles
To get a container into a stacking bay, elevating transfer vehicles are used **[5]** as shown in Figure 4.9. These vehicles take care of the vertical transportation and, when needed, the horizontal transportation in the warehouse. An elevating transfer vehicle comprises of a stiff framework attached to a rail mounted support vehicle. In the framework a cradle **[10]** can be lifted and lowered as desired. The cradle contains a shuttle **[11]** that can be moved laterally in and out of the cradle to extend into a bay in the storage rack or overhead portal crane area.

For example: as the elevating transfer vehicle arrives at the storage rack, the shuttle leaves the cradle and enters a bay containing a container **[7]**. Having located itself under the container, the shuttle lifts the container from the four lower corners supporting devices **[12]**. The shuttle returns to the cradle of the elevating transfer vehicle **[10]**, which will move itself to the desired new container location where the process will take place in reverse order.

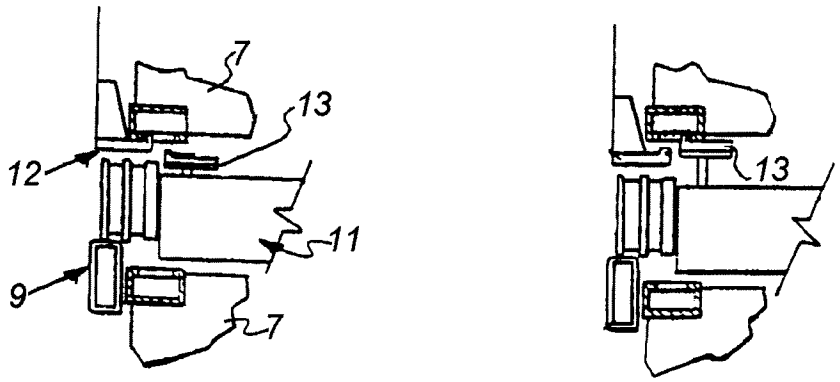


Figure 4.10, front view of the lifting process of a shuttle

In Figure 4.10 the lifting process of a shuttle is shown. The shuttle **[11]** releases the container **[7]** from its four supporting devices **[12]** by lifting it with a lifting device **[13].**

In Figure 4.11 the cradle **[10]** of the elevating transfer vehicle is shown. The container **[7]** is supported by the shuttle of the elevating transfer vehicle **[11]**, which in turn is supported by the underside of the cradle **[6]**.

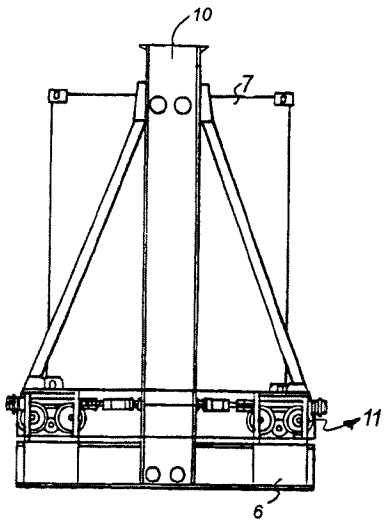


Figure 4.11, the cradle of the elevating transfer vehicle

4.2.2 Adaptations to the chosen warehousing concept

In the previous chapter the chosen container-warehousing concept has been described. The choice was based on the random container access and the absence of the container stacking height limits posed by the ISO standards. However, there are some apparent shortcomings. These shortcomings will be described and addressed in this chapter.

- Stack production
  As can be seen in the top view in Figure 4.8 the number of portal cranes and unloading/loading platforms **[1]** are limited. As a result, bottlenecks might appear during the handling process.

- Separation of functions on the ground level
  There are four functions present on ground level as can be seen in Figure 4.7. The first function is horizontal transportation consisting of horizontal transportation lanes **[4]** and horizontal transportation units **[2]**. The second function is formed by the unloading/loading areas where the portal cranes **[1]** operate. The third function is the area reserved as the operating lanes for the elevating transfer vehicles. This arrangement will create problems at possible crossing points between horizontal transportation units and elevating transfer vehicles. The fourth and last function on the ground level is container storage. As the necessity for container storage on the ground floor is limited, this area might have a better application. Container storage bays might just as well start at the second floor.

- Land usage
  As land usage is to be minimised in this study, some additional adaptations have to be made. In the container storage area on the container terminal, the object is to design container storage bays without unused area on the higher floors. Currently, the area above the horizontal transportation lanes **[4]** is left unused as can be seen in Figure 4.7.

The adaptations applied to the container warehousing system necessary to solve the problems posed above are discussed in the following.

Unloading/loading area
The changes in the warehousing concept are most apparent for the unloading/loading area as can be seen in Figure 4.12. By moving the container storage bays to the second floor of the warehouse a great deal of space can be gained on the ground floor. This enables the placement of two extra horizontal transportation lanes **[14]** to be placed thereby considerably increasing the stack production.

The most striking difference however, is the absence of the portal cranes. The portal cranes have been replaced by the so-called service points **[15]**. A service point consists of one container location located upon an elevated area with the same height as a horizontal transportation unit **[2]**. In order to make the lateral horizontal movement during the unloading or loading of a container upon a horizontal transportation unit, each service point is provided with a shuttle **[11]** comparable to the shuttle of the elevating transfer vehicle.

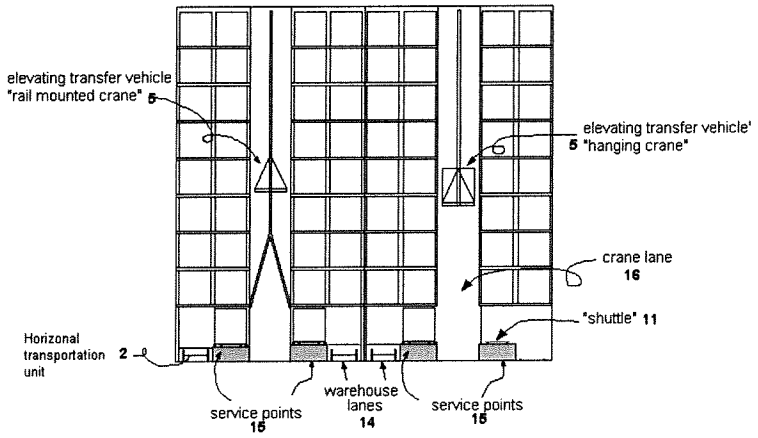


Figure 4.12, a cross section of two warehouses

In Figure 4.13 the top view of the adapted warehouse can be seen. The two extra horizontal transportation lanes **[14]** can be distinguished as well as the locations of the service points **[15]**. The service points are all coloured grey and are distributed all along the horizontal transportation lanes. This will secure a great increase of container handling production.

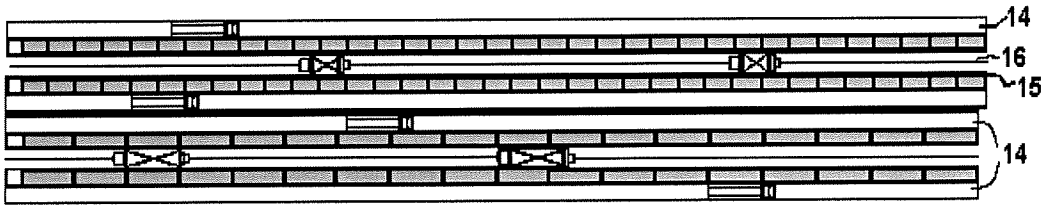


Figure 4.13, top view of the warehouse with adaptations

The loading procedure of a container located on the shuttle in the cradle of the elevating transfer vehicle onto a service point is as follows: when the elevating transfer vehicle arrives next to the service point, the shuttle in the cradle containing the container transfers lateral onto the service point. The cradle shuttle lowers the container onto the service point shuttle and returns back to the cradle allowing the elevating transfer vehicle to proceed with its next task. As a horizontal transportation unit arrives at the service point requesting a container, the service point shuttle lifts the container from the supporting devices and transfers lateral upon the horizontal transportation unit where it lowers the container onto supporting devices. The shuttle then transfers back onto the service point and waits for the next task.

The great advantage of this system is that a horizontal transportation unit can be serviced while an elevating transfer vehicle is still busy on an other job. This increases the flexibility of the system.

Elevating transfer vehicle
In Figure 4.12, two possible types of elevating transfer vehicles are shown. The rail mounted crane and the hanging crane. The ground floor has been kept free in order to allow the crane to cross any horizontal transportation lanes, which increases the flexibility of the concept.
Of the two described crane possibilities the rail-mounted variant seems be in favour. A hanging crane could also be a possibility, but it will be much harder to make the construction stiff and rigid. This is necessary to avoid time loss due to sway problems.

4.3 Horizontal transportation

There are numerous horizontal transportation concepts. The choice of a horizontal transportation concept largely depends on the type of terminal that is to be designed. A terminal with relative low container throughput and a high availability of terminal area will have an other optimal horizontal transportation concept than a terminal with relative higher container throughput and a lesser availability of terminal area. The choice also depends on the suitability to work together with the two other terminal components portainer and container storage.

The terminal in this study will be designed as an automated high production terminal while minimising the terminal area. The horizontal transportation concept will have to comply with these specifications resulting in the following demands.
The horizontal transportation units have to preferably:

- be completely automated
- be able to process large amounts of containers
- occupy a minimum of land area

Taking the automation demand into account the options are limited to automated guided vehicles. The most credible ones are:

- Rubber tired automated guided vehicles (agv's)
- Rail guided automated guided vehicles (rgv's)

In the next sections the both horizontal transportation concepts will be illustrated and a choice between them will be made.

4.3.1 Rubber tired automated guided vehicles (agv's)

Agv's have already been fully operational for over a longer period of time on the ECT-terminal in the port of Rotterdam. Due to the operational experience, the agv has gained in reliability. Within the FAMAS-study, research has also been carried out on improvements of the ECT-agv's [L.11], resulting in the FAMAS agv.

Driving force
Each agv has a hydrostatic diesel engine that is responsible for the propulsion. The agv performance is the same for normal and reverse driving. The disadvantages of this kind of propulsion are the refuelling and maintenance stops. Refuelling stops are scheduled once every 80 hours during normal operation [L.12].

Navigation
The navigational system at the ECT terminal consists of a line grid of inductive wires orientated parallel and perpendicular to the quay wall, called the Free Ranging on Grid (FROG) system [L.10]. Sensors detect possible deviations in the path of the agv. The deviations are then registered and compensated by the agv. In this manner a positioning accuracy of ± 3 cm can be achieved.

Operational properties
In table 4.1 the properties of the ECT-agv and of the FAMAS-agv are given [L.12]

Operational properties	ECT-agv	FAMAS-agv
Speed loaded (m/s)	5.5	7.0
Speed unloaded (m/s)	4.0	7.0
Acceleration loaded (m/s^2)	>0.3	0.5
Acceleration unloaded (m/s^2)	>1.0	>1.0
Speed in turns (m/s)	3.0	>3.0

Table 4.1, operational properties of a rgv

As land usage is an important design parameter in this study, the required turning radius of an agv must be minimised. A small turning radius can only be established with low agv speed in order to achieve the maximum wheel turning. Wheel turning speed of the inside wheel of an ECT-agv is 15° per second. With the agv speed decreasing to 3 m/s the turning radius is 18 meters. The time needed to complete a 90 degree turn with this configuration is 9.5 seconds.

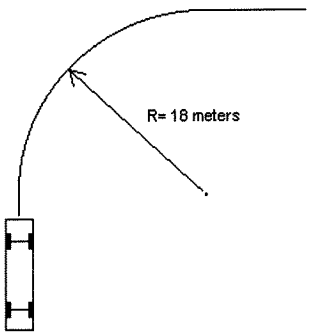


Figure 4.14, agv turning radius

4.3.2 Rail guided vehicles (rgv's)

Another system for horizontal transportation is rail guided agv's or rgv's. Preussag Noell is developing this system in Germany and in the People's Republic of China [L.13]. Based on technology and experience acquired during the exploitation of the transrapid magnet trains, the rgv concept has been specially adapted for high performance terminals. The layout of the rails consists of runways orientated parallel and perpendicular to the quay suitable for longitudinal and lateral displacements of rgv's. The wheel sets of the rgv can rotate trough 90° angles as can be seen in Figure 4.15.

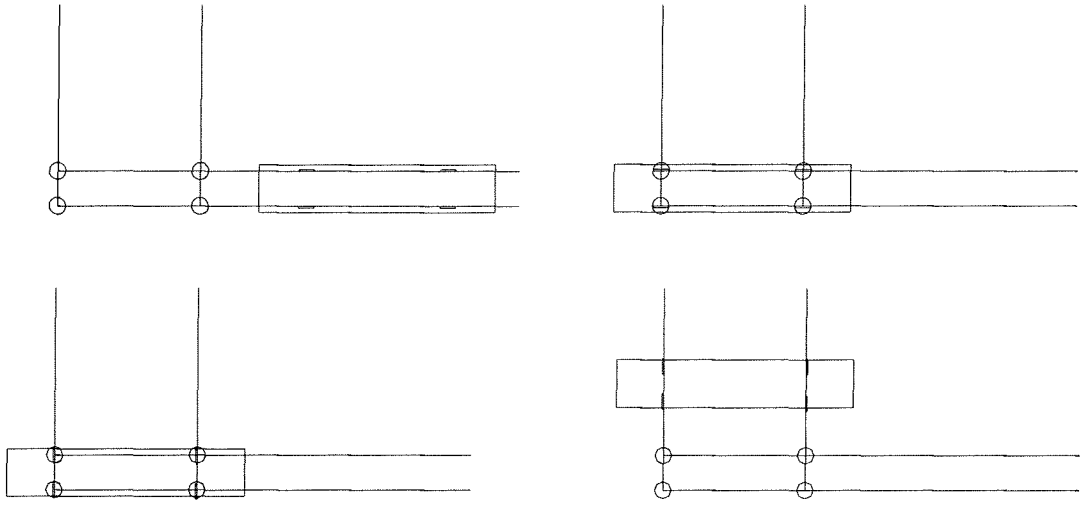


Figure 4.15, rgv taking a turn

The runway consists of ordinary UIC 60 rails mounted on steel twin sleepers. The rails must be accurately laid to guarantee a gap between magnets and the stator of 7 ± 3 mm. The guide system consists of three modules: a motion for transversal transition, a module for longitudinal transition and the crossings where the two directions meet. To make it possible to turn all four wheel sets of the rgv's, four circular steel discs with transverse guides have been fitted at the crossing points. A hydraulic power unit turns the discs individually but simultaneously.

Driving force
The rgv's are driven by means of contact free linear synchronous motors, which are distributed over the layout as required for a suitable driving force. They act on the permanent magnets located on the underside of the rgv. It is possible to set a variable speed by means of a mobile electromagnetic field, generated by a frequency converter.

Navigation
A contact free actual detecting system is integrated into the runway, responding to the individual magnets located on the rgv. This allows to determine the position of the rgv and to supply the input values required to ensure that the linear drivers are supplied with power and switched over in the correct order. The claimed position accuracy is ±3 mm.

Operational properties
The operational properties of the rgv during tests on the demonstration plants of Preussag Noell have been taken from [L.13] and are shown below.

Operational properties	rgv
Speed loaded (m/s)	3
Speed unloaded (m/s)	3
Acceleration loaded (m/s^2) (Assumed property)	1
Acceleration unloaded (m/s^2) (Assumed property)	1
90°-wheel rotation (s) (Assumed time)	4

The rgv has a length of 15 meters and is 2.8 meters wide.

4.3.3 Criteria and choice

The choice which horizontal transportation concept to use for the design of the terminal is based on a number of criteria. These criteria will be discussed in this section.

Operational properties
Operational properties like speed and acceleration will not be the most important criteria for the choice of the horizontal transportation system, as the properties are more or less comparable for both horizontal transportation units. Especially because the terminal will be designed as a compact terminal driving distances will be limited.

The time needed to complete a 90° turn is comparable for both alternatives. A rgv travelling at a speed of 3 m/s consumes 10 seconds for deceleration (3 s), wheel turning (4 s) and acceleration (3 s) while the agv consumes 9.5 seconds for completing the turn.

Land usage
The land usage of the horizontal transportation vehicles is an important planning factor. During the taking of the turns the land usage of the rgv's is more efficient than that of the agv's. This results in a better use of the area on the terminal. The land usage of the horizontal transportation units is the key choice factor in this study as land usage an important objective.

The preferred horizontal transportation unit type for the design of the terminal is the rgv. The most important criterion is the usage of terminal area.

5 The design of the terminal Layout

In this chapter the layout of the container terminal will be established. The layout consists of rgv lanes, portainers and stacks. The realisation of the layout will be reproduced step by step thereby giving an impression of all the design considerations that have been made throughout the designing process.

First, the capacity and the positioning of the container warehouses will be dealt with, followed by the positioning of the two types of rgv lanes in the layout. Some preliminary layout calculations have been made in order to get a first impression on the capacities of the terminal.

5.1 Positioning of the container warehouses

As the terminal will not be implemented in a specific topographical environment, the placement of the container warehouses will be assumed not to be affected by geological restrictions. The warehouses will therefore be placed directly behind the quay containing the portainers.

As stated in Chapter 3, the quay of the terminal will contain three longitudinal orientated adjoining berths. By placing the warehouses directly behind the portainers, the orientation of the warehouses will be determined by the properties of the rgv's. The warehouses have been designed in such a way that rgv's can only enter in a straight on fashion. The consequence for the warehouses is that they are to be orientated parallel to the quay as can be seen in Figure 5.1.

The next step in the design process is the determination of the required storage capacity per yard behind each berth, for the pure transhipment terminal. The capacity calculation has been carried out by assuming arrival cycles of feeders and mega vessels.
Based on the arrival cycles of the vessels and the amount of containers unloaded and loaded per call, the needed total storage capacity of the three yards has been calculated in Chapter 3.

The required container storage capacity is 24,000 TEU. The storage capacity will be spread out evenly over the terminal. Berth 1 and Berth 3 both have a length of 250 meters while Berth 2 has a length of 370 meters. Proportionally Berth 1 and Berth 3 must each be able to host an amount of 6,860 TEU, while Berth 2 should be able to host an amount of 10,290 TEU.

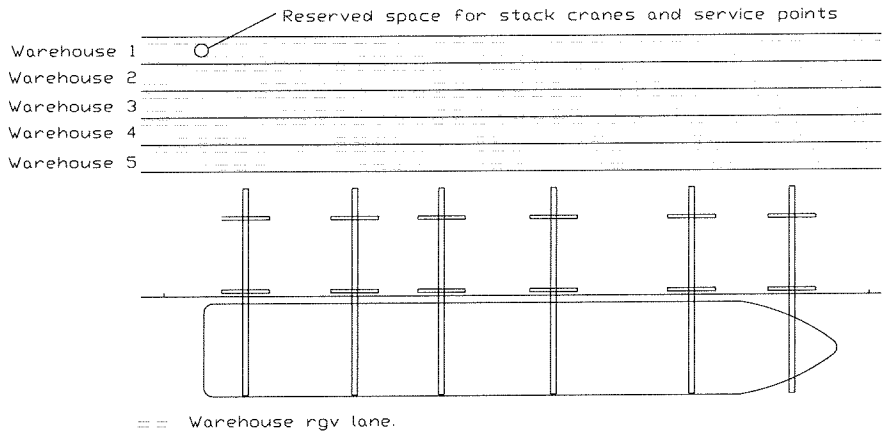


Figure 5.1, first design phase

Based on Appendix 5 the number of warehouse rows has been calculated to amount to 5, and will be placed over the length of the terminal. By choosing 5 warehouse rows, the stacking height will amount to 12 stories of warehouse racks. Adding the ground level means that the warehouse will be 13 stories in height. It is assumed that it is possible to construct a feasible container warehouse with this height. However, it is recommended to conduct a study on this subject.

The length of a container has great influence on the dimensions of the storage bays and the warehouse cranes. For this reason 20' container stacks have to be separated from the 40' container stacks. Having assumed a TEU factor of 1.5 container storage locations must be consistent with this relation. This means that the number of 20' container bays and 40' container bays should be the same. As one 20' container warehouse contains twice as much storage bays as a 40' container warehouse, the ratio 20' warehouses – 40' warehouses should be 1:2. Having implemented 5 container warehouses, the best approach to the 1:2 ratio is by reserving 2 warehouses for 20' containers and 3 warehouses for 40' containers. Warehouse rows 1 and 2 will therefore be 20' container warehouses, while warehouse rows 3 to 5 will be designed as 40' container warehouses.

5.2 Positioning and capacity of the rgv lanes

Having established the properties of the yard in the previous section the layout will now be expanded with the rgv lanes. Besides the warehouse lanes inside the warehouses, yard-lanes and inter-berth lanes can be distinguished. Yard-lanes are orientated perpendicular to the quay and guide the rgv's coming from outside the yard into the yard. Inter-berth lanes are orientated parallel to the quay and maintain the connection between the berths. Placing these lanes on the terminal will form a grid of lanes. In the following chapters the required capacity of both lane types will be calculated followed by the placement of the lanes.

5.2.1 Inter-berth lanes

In order to calculate the required number of inter-berth lanes the capacity of one single lane must be known first. In Appendix 6 the capacity of an inter-yard-lane has been determined to be about 4 rgv's per minute or one every 14.3 seconds. The normative situation to be used for calculating the needed lane capacity will be adjacent loading on all three berths.
This means that two feeder vessels moored at Berth 1 and Berth 3 and a mega vessel moored at Berth 2 are being loaded as explained in Chapter 3.5. Figure 5.2 shows the schematised situation.

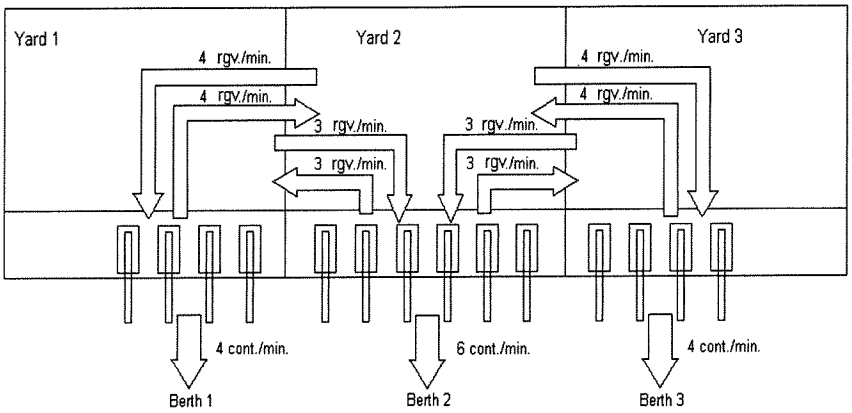


Figure 5.2, schematisation of inter-berth rgv flow when three vessels are being adjacent loaded.

When a mega vessel is adjacent loaded, portainer production is in average one per minute. As the containers required for the loading on Berth 2 are assumed to be equally distributed over Yard 1 and Yard 3, 3 containers per minute should be provided from these two adjacent berths to the portainers located on Berth 2. The same amount of empty rgv's will leave the portainers to collect a next container. This means that 3 rgv's will pass both borders of Yard 2 in two directions every minute.

The rgv's working for the portainers located on Berth 1 and Berth 3, will all collect their containers from Yard 2 during adjacent loading. Four rgv's a minute will pass the borders between Yard 1 and Yard 2 in both directions. The same amount will pass the borders between Yard 2 and Yard 3 every minute. Adding up the inter-yard container flows shows a needed rgv capacity of 7 containers per minute in both directions and on both boundaries. This results in 14 rgv boundary crossings per minute.

The capacity of an inter-yard-lane is 4 rgv's per minute as is illustrated in Appendix 6. With 4 inter-berth lanes the inter-berth transportation capacity is about 16 rgv boundary crossings a minute. This seems to be enough, but during the simulation phase the rgv flows will probably become more irregular. Applying more inter-berth lanes will increase the flexibility of the inter-berth transportation. This is the reason that the simulation study will be carried out with 8 inter-berth lanes as can be seen in Figure 5.3. Later on a simulation run with 4 inter-berth lanes will be carried out to check this assumption.

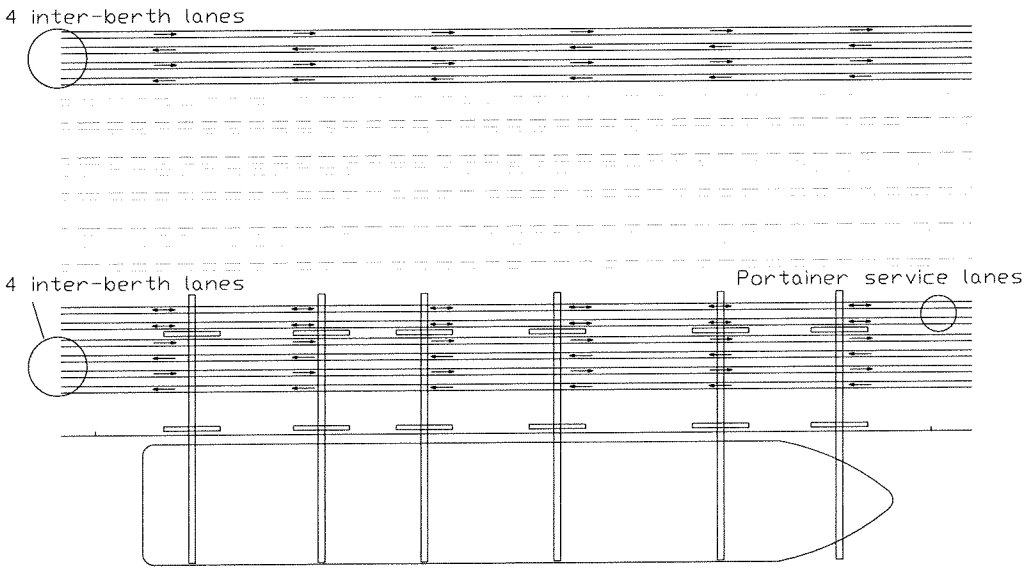


Figure 5.3, Layout of Berth 2 including inter-berth lanes

Arrows represent the lane directions. The portainer service lanes have been placed behind the portainers. For now it is assumed that two will suffice. The computer simulation will give verification of this assumption.

5.2.2 Yard-lanes

The rgv lanes that make the warehouses accessible are called the yard-lanes. These lanes connect the warehouse lanes to the inter-berth lanes and the portainer service lanes. The yard-lanes will be placed perpendicular to the quay in the layout. The number and distribution of the yard-lanes depend on the required lane capacity. The lane capacity of the yard-lane is calculated in Appendix 6 and about 4 rgv's per minute.

Having established the yard-lane capacity makes it possible to place the yard-lanes in the layout. The average portainer production is again assumed to be 1 container per minute. This implies that during the unloading/fast-loading of a vessel, four portainers can be connected to one yard-lane. However, if during the unloading/fast-loading of a vessel another vessel is being adjacent loaded, the needed capacity of the yard-lane will increase dramatically. For this reason the yard-lanes have been designed to be supplied by two portainers. Whether or not the yard-lane has enough capacity to be handled by two portainers, has been checked in Appendix 7 by performing a performance calculation for the entire terminal.

During the unloading/fast-loading of a vessel, the rgv's cover a loop between the portainers and the stacks located in the yard. There are two distinctive variations for the placement of the yard-lanes. These two variations will be discussed now.

Opposite rotation rgv loops
In this case two adjacent portainers have opposite orientated quay-stack loops, as can be seen in Figure 5.4. As one yard-lane can handle a capacity of at most two portainers every portainer is enclosed by a rgv loop. This way a tight grid can be formed. Each warehouse lane now holds three 40' container or six 20' container service-points. The warehouse lanes are capable of hosting a maximum of 3 rgv's.

In order to make it possible to shuffle containers more efficiently between berths, half of the warehouse lanes located behind a portainer have been given an opposite direction to the described unloading loop. Rgv's carrying out shuffling orders can also enter the stack from the above-situated inter-berth lanes.

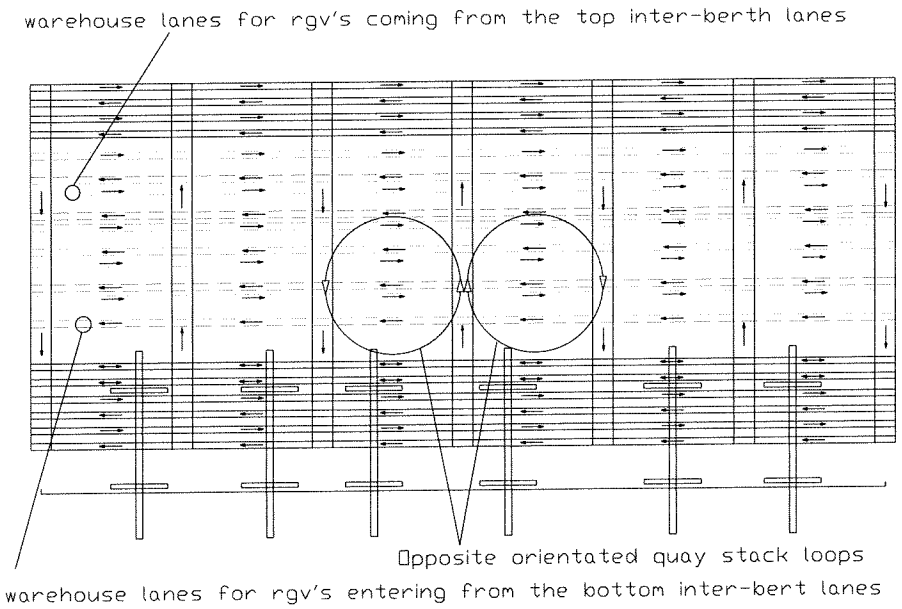


Figure 5.4, Layout version 1 including warehouse lanes

Equal rotation rgv loops
As is the case in the previous layout, one warehouse lane connects two adjacent portainers to the stacks located behind them. The circular motion of the rgv's is now equally orientated instead of opposite resulting in the layout designed in Figure 5.5.

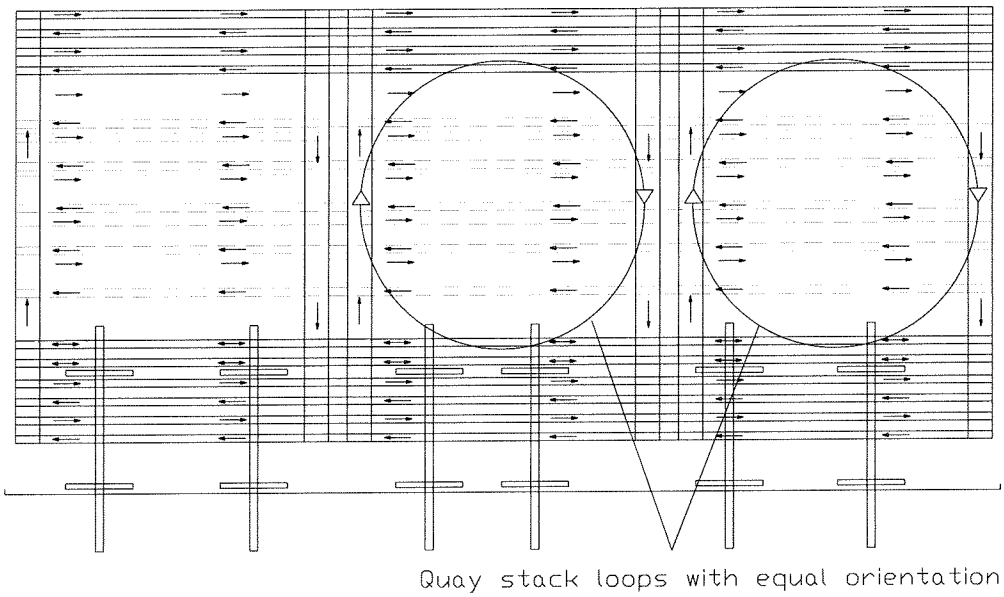


Figure 5.5, layout version 2 including warehouse lanes

The number of 40' container service points per warehouse has increased to 6, while the number of 20' container service points has increased to twelve per warehouse.
The grid of the version 2 layout is less tight than the version 1 layout, resulting in a higher chance of congestion. Another problem might appear due to the fact that the yard lanes are placed right next to each other. Portainers located above the two yard-lanes might have difficulties in reaching the rgv's waiting in their queue while handling a vessel.

The tighter grid of version 1 combined with the possible handling problems of the portainers in equal rotation are the reasons the computer simulation will be carried out with the opposite loops layout.

In Appendix 7 the overall needed capacity of the established layout has been calculated for the most extreme situations. The capacity seems to be sufficient on all the lanes. The computer simulation will give a more definitive answer.

5.3 Additional terminal design aspects

Some terminal aspects have not yet been included in the terminal layout and will be briefly clarified in this section.

Rgv-parking
Not all available rgv's will be active constantly. For this reason a rgv-parking has been added to the layout. The positioning of the parking has not been extensively examined, but can be positioned behind the stacks past the inter-berth lanes, as can be seen on Figure 5.7.

Hatches of the container vessels
A number of container vessels are equipped with hatches that separate the containers located in the hull of the vessel and containers located on top of the vessel. These hatches need to be removed and placed on the quay during the handling of the vessel. A space has been reserved behind the portainers.

Elevated rear rails of the portainers
In this study only the processes on the terminal will be simulated during the handling of container vessels. A result of not including vessel arrival schemes makes the determination of the ideal number of portainers impossible. For this reason the layout drawings of the terminal always include the maximum number of portainers. In a later phase however an optimum number of portainers should certainly be determined. This probably means that the number of portainers will be less than the maximum number. This implies that portainers have to be able to transfer from one location to another. Yard lanes will have to be crossed during this process. As the rail mounted hind legs of the portainers may not cross the rgv yard lanes the hind legs of the portainer will be elevated to the first floor allowing the rgv's to pass underneath.

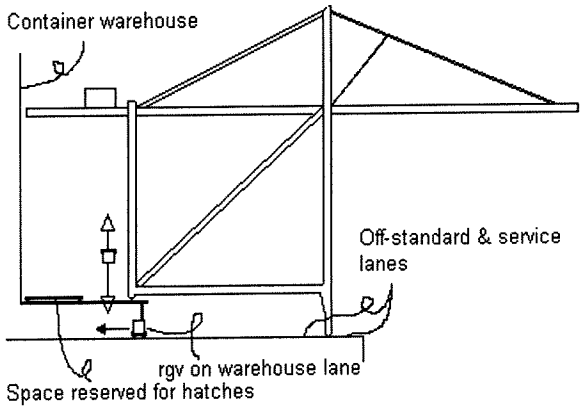


Figure 5.6, side view of the portainer

Service lanes and off-standard lanes
Service lanes will be placed in front of the portainer, while the off-standard lanes are placed behind the front legs of the portainers.

In the previous sections of this chapter the evolving of the layout of the terminal has been described one step at a time. Integrating the chosen framework of rgv lanes, the container storage and the portainers on their portainer locations, will give a complete overview of the terminal. In Figure 5.7 all the items on Berth 1 have been named.

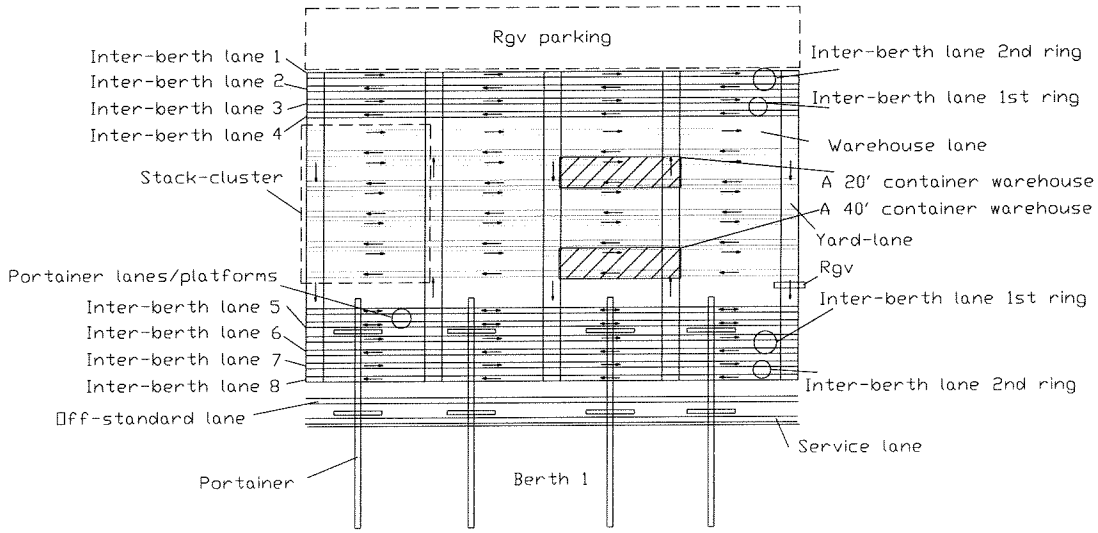


Figure 5.7, Berth 1

The complete terminal layouts have been included in the appendices. The first layout as show in Appendix 1 contains the dimensions, while the second layout, shown in Appendix 2, shows the numbers given to all the nodes. The numbers of the nodes will be used to simulate the terminal.

5.4 Handling strategies applied to the terminal Layout

The handling strategies as described in Chapter 3 will now be applied to the chosen layout. The chosen handling strategies for the three berths consist of unloading, fast loading, adjacent loading and shuffling. In this order the strategies will be dealt with in this chapter.

5.4.1 The unloading of a container vessel

When a container vessel is being unloaded the containers will always be delivered to one of the stacks located in the stack-clusters behind the unloading portainer. This unloading strategy is the same for feeder vessels moored at Berth 1 or Berth 3 and for the mega vessels moored at Berth 2. The rgv's delivering the containers from their portainer to a stack will always drive in the same circular motion. The direction of this circular motion depends on the location of the unloading portainer. A consequence of the fixed direction of the circular motion is that the unloading rgv's can only enter 5 of the 10 warehouse lanes of a stack-cluster, as these lanes are one directional, as can be seen in Figure 5.8.

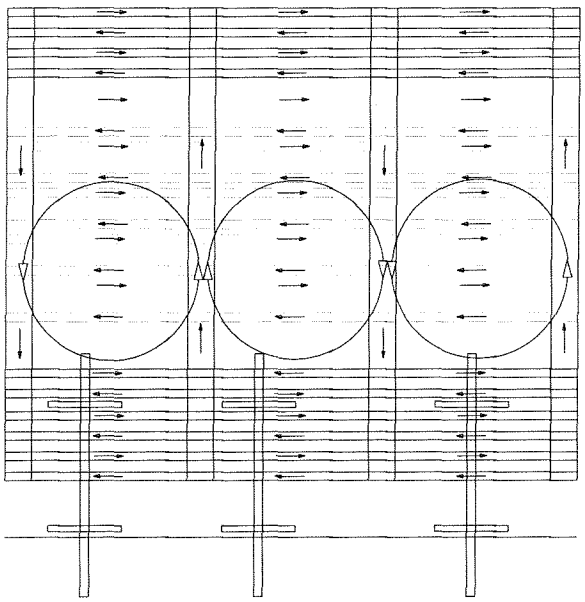


Figure 5.8, unloading strategy of a container vessel

The unloading rgv's will not enter an inter-berth lane during the unloading process. A consequence of this strategy is that the yard-lanes will be more crowded close to the portainer, as all of the unloading rgv's will have to enter the portainer queues. Closer to the far end of the yard, opposite of the portainer, the yard-lanes will be less crowded as rgv's have already entered warehouse lanes located closer to the portainers. Rgv's arriving from Berth 2 might make use of this.

5.4.2 The adjacent loading of a mega vessel

As opposed to the unloading strategy, which is the same for all berths, the adjacent loading strategy differs per berth. First the adjacent loading strategy of the mega vessels will be dealt with. The mega vessel being handled is always moored at Berth 2. After having been unloaded the mega vessel can be loaded in three different manners. The mega vessel can be adjacent loaded, which means that the to be loaded containers have to be retrieved from the yards in which they have been unloaded into by the feeder vessel berths. A second possibility is that the to be loaded containers have already been shuffled to Yard 2 from the feeder vessel yards, and can be loaded from Yard 2. This is called fast loading. The third option is a combination of the two loading strategies. As the fast loading of a container vessel follows exactly the same pattern as during unloading the fast loading strategy will not be further explained.

Because the feeder vessels provide the terminal with containers for the mega vessels, the adjacent loading strategy of Berth 2 consists of retrieving the to be loaded containers from Yard 1 & 3 and delivering them to the portainers located on Berth 2. The rgv's will take the inter-berth lanes during this strategy. In order to spread out the rgv flow, it has been split in a rgv flow that make use of the top inter-berth lanes and into a rgv flow that makes use of the bottom inter-berth lanes. In the simulation program the even numbered rgv's take the bottom inter-berth lanes, while the odd numbered rgv's take the top inter-berth lanes. Two example routes have been drawn in Figure 5.9, one route for an even numbered rgv and one route for an odd numbered rgv.

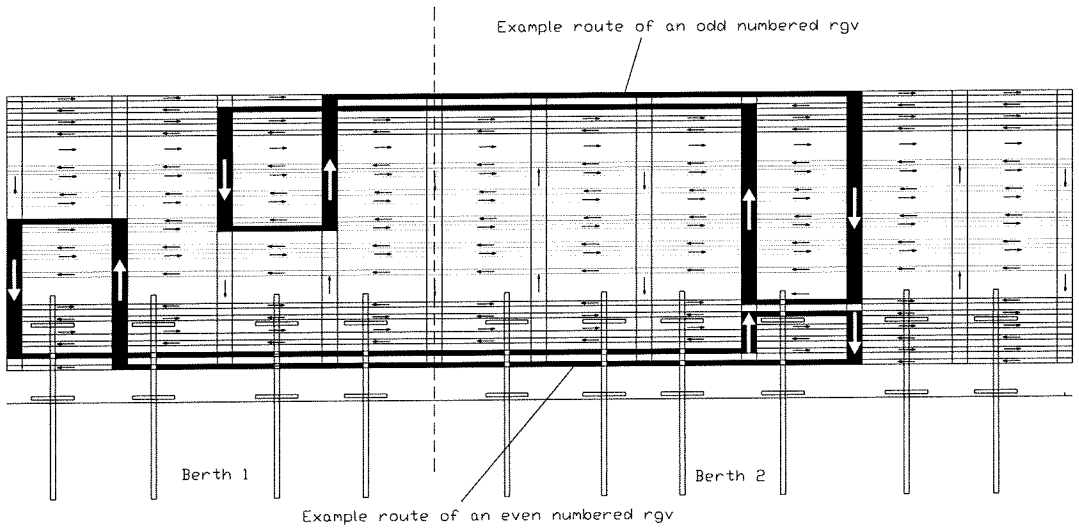


Figure 5.9, adjacent loading from Yard 1 to Berth 2

During the adjacent loading of Berth 2 the to be loaded containers will, of course, also be loaded from Yard 3. As can be seen in the above figure, the rgv's working for the portainers located on Berth 2 will take the outer "ring" of inter-berth lanes. The inner "ring" is reserved for rgv's from the feeder berths 1 & 3. This can be seen in the next part of the chapter.

5.4.3 The adjacent loading of a feeder vessel

As is the case with a mega vessel the situation can occur that the terminal has not had enough time to shuffle the containers to the Yards 1 & 3. In that case the containers have to be adjacent loaded instead of being fast loaded. During the adjacent loading of a feeder vessel, the rgv flows will also be split up. An example of two rgv routes for the adjacent loading of a feeder vessel moored at Berth 1 can be seen in Figure 5.10.

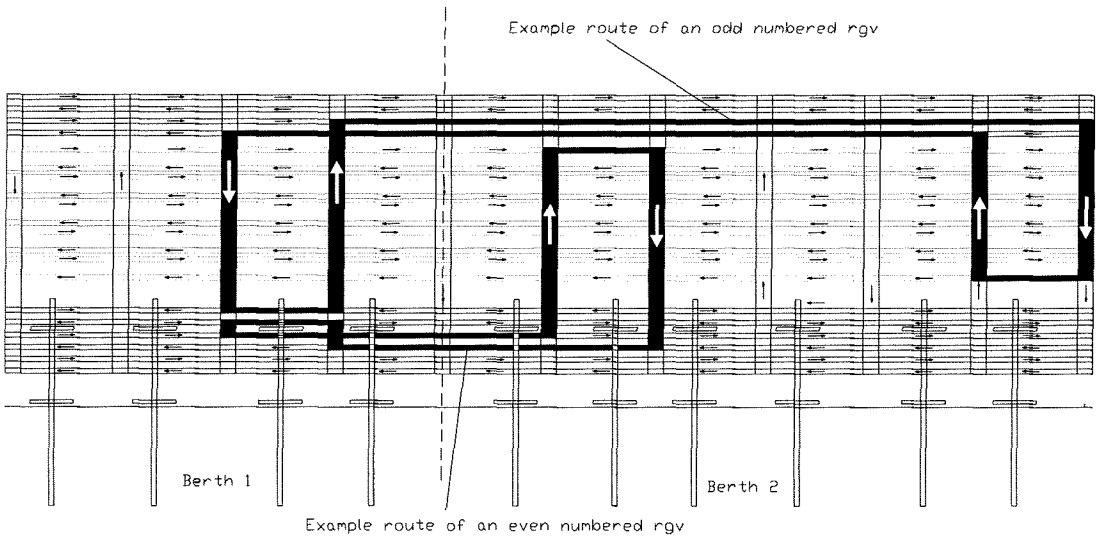


Figure 5.10, adjacent loading of a feeder vessel from Yard 2

During the adjacent loading of the feeder vessel, the rgv's will take the first "ring" in order to get to Yard 2.

5.4.4 The shuffle process on the terminal

Shuffling from Yard 2 to Yard 1 & Yard 3

When a mega vessel has unloaded its containers in Yard 2 the shuffling process on the terminal will begin when the mega vessel has been loaded and has just left the terminal. The rgv's will not be sent back to the parking but will remain working as shuffle rgv's. As the portainers of the berth will be inactive during the shuffling process the rgv's will be subscribed to the stack-cluster located behind the portainers. Each time a container has been shuffled to Yard 1 or 3 the rgv will return to this stack-cluster in Yard 2 in order to retrieve the next shuffle container.

As is the case with adjacent loading for Berth 2, the rgv routes will be split up in two. The top routes will be reserved for the odd numbered rgv's, while the bootom routes will be reserved for the even numbered rgv's. The shuffling rgv's will also take the second inter-berth ring, as is the case during adjacent loading for Berth 2.

A typical shuffle procedure is as follows. A rgv has just delivered one of the last containers for during the loading of a mega vessel. The rgv will be given the assignment to start shuffling. The rgv will drive to a stack behind the portainer with a yard lane orientated in such a way that the rgv can enter the correct inter-berth lane, as it is dependent on its number being odd or

even. After the warehouse crane has loaded the rgv, it will shuffle the container to Yard 1 or Yard 2 where it will enter the right stack. The container will be unloaded from the rgv by the warehouse crane and depending if a container has to be shuffled back to Yard 2 it waits to be loaded again. The rgv loaded or not will return to a stack located in its home stack-cluster located in Yard 2.

The rgv will actually take comparable routes as during adjacent loading. The only difference is that the rgv's will not be handled by their portainers but by on the five stack-cluster warehouse cranes.

Shuffling from Yard 1 or Yard 3 to Yard 2

When a feeder vessel has been unloaded and loaded again the containers deposited in Yard 1 or Yard 3 will be shuffled to Yard 2. Like during the adjacent loading of a feeder vessel the shuffling rgv's will take the first inter-berth lane ring.

5.5 Number of warehouse cranes

In the chosen layout five warehouses have been placed behind each portainer. As has been stated before, a warehouse hosts two warehouse lanes. Each of these lanes is equipped with a row of three service points for 40' container stacks or six service points for 20' container stacks. An initial assumption has been made that each warehouse hosts one warehouse crane. This brings the total amount of warehouse cranes to 14 * 5 = 70 warehouse cranes. Having assumed this, some initial occupancy rate calculations have been made for three situations.

During the unloading/fast-loading of a container vessel the rgv's assigned to a portainer only deposit and retrieve their containers to and from the stacks located behind the portainer. Given the average portainer production of 1 container per minute the occupancy rate of the five stack-cranes has been calculated in Appendix 9. These unloading/fast-loading calculations have been made without interference of rgv's engaged with adjacent loading of other berths as can be seen in Figure 5.11.

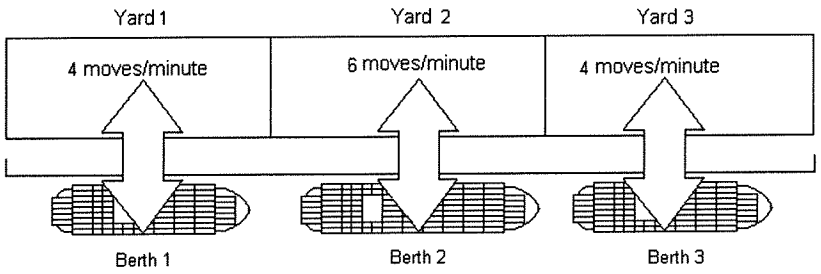


Figure 5.11, portainer production during unloading/fast-loading

Having the portainers located on Berth 1 unloading/fast-loading a vessel, can increase the occupation rates of the warehouse cranes. By having a vessel adjacent loading on Berth 2, the occupation rates of the warehouse cranes located in Stack 1 will increase. The situation is illustrated in Figure 5.12.

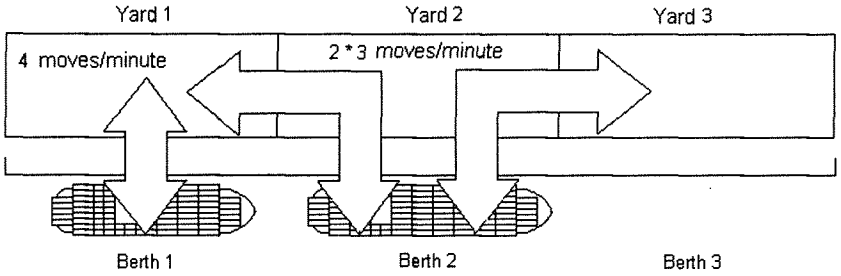


Figure 5.12, Portainer production during unloading fast loading on Berth 1 and adjacent loading on Berth 2

An even busier situation can be created when the warehouse cranes located behind berth 2 not only provide the unloading/fast-loading rgv's of the own portainers but also provide the rgv's arriving from the neighbouring berths 1 and 3, during adjacent loading. These calculations have also been made in Appendix 9.

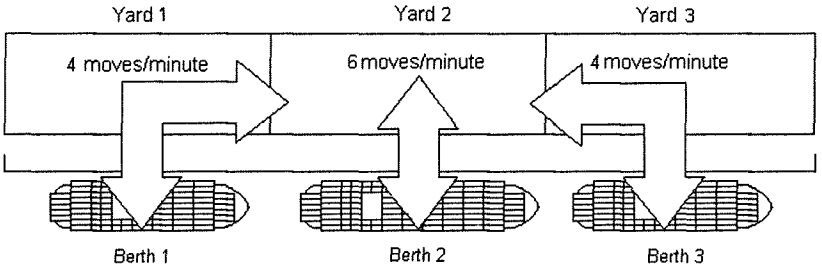


Figure 5.13, portainer production during unloading/fast-loading on berth 2 and adjacent loading on berth 1 and berth 3.

The results of the heavy occupied and the less heavy occupied situation have been summarised in Table 5.1.

Stack location	Stack carne occupancy in stacks behind Berth 1		Warehouse crane occupancy in stacks behind Berth 2	
Situation	Normal occupancy (Figure 5.7)	Heavy occupancy (Figure 5.8)	Normal occupancy (Figure 5.7)	Heavy occupancy (Figure 5.9)
20' container stack	33.3 %	58.3 %	33.3 %	77.8 %
40' container stack	22.2 %	39 %	22.2 %	51.9%
Combined	27%	49 %	27 %	64.8 %

Table 5.1, Warehouse crane occupancy rates

The warehouse cranes located in the warehouses behind Berth 2 have a maximum occupancy rate of 77.8 % during the unloading/fast-loading of a vessel moored at Berth 2. At the same time rgv's subscribed to portainers located on Berth 1 and Berth 3 are also retrieving containers from the stacks behind Berth 2. This occupancy rate is rather high considering the calculation is an average occupancy rate. With the computer simulation the results of this situation can be analysed more thoroughly. If the capacity of the warehouse cranes will show to be insufficient, warehouse cranes located in the stacks behind Berth 1 and Berth 3 may transfer to Berth 2 in order to increase the stack capacity.

5.6 Number of rgv's on the terminal

The number of used rgv's during a certain procedure will depend on the length of the mean transportation cycle, the portainer production rate and the properties of the rgv, like speed and acceleration. As the transportation cycle length is the most important variable, a calculation has been made for the main transportation cycle length belonging to each handling procedure on the terminal.

The mean transportation distance during the unloading of a vessel is shorter than the mean transportation distance during adjacent loading. In Appendix 10 the mean transportation distances belonging to the different handling strategies on the terminal have been calculated. Based on the mean portainer handling times and the speed of the rgv's on the different lanes the numbers of needed rgv's have been calculated. The results are given in Table 5.2.

Procedure	Mean transportation cycle time (seconds)	Number of needed rgv's (rounded off)
Unloading a vessel	248	5
Adjacent loading	529	9
Shuffling	551	9

Table 5.2, number of needed rgv's per portainer

The number of rgv's that will be used during the computer simulation is 1 above the calculated needed number. Rgv's will almost certainly meet each other on crossings resulting in waiting queues on certain nodes. In order to keep the productivity of the portainers high 1 extra rgv might even be the minimum. The computer simulation will start off by using 6 rgv's during unloading and 10 rgv's during adjacent loading and shuffling.

6 Computer simulation

6.1 Problem analysis tool: PROSIM

The designed container terminal will be simulated in PROSIM. PROSIM is a process description method language. When using such a method the model can be observed as a number of components interacting with each other in such a way that they reflect the actual system as close as possible. Building a model means identifying the components and describing their activities in their own module. A computer simulation model will thus consist of different modules, which are connected by the fact that the components interact with each other. These interactions are described in the modules of the components.

A component can either be permanent or temporary. A permanent component will be present in the model permanently whereas a temporary component will only be present for a certain period of time. Every component has certain characteristics that are important for the proper functioning of this component and for the interacting with other components. These characteristics are so called "attributes". When components have the same type of attributes they form a "class of components".

A Prosim model consists of two sections:

- A definition section in which all the components and their attributes, sets and other variables are defined. This section is called the DEFINE module;
- A dynamic section in which the dynamic behaviour of the components is described. This section consists of "modules" and "macros".

A PROSIM model always starts with a module called MAINMOD or Main. This module takes care of the run control and the creation and activation of the first components. The other modules are activated either by the Main module or by another module.

A module can activate a macro. A macro contains a description of certain activities to be executed. These activities are repeated regularly by the module and are therefore put in a macro in order to keep the module as simple as possible.

6.2 Definition of the components of the computer model

The model consists of a number of components interacting with each other in such a way that they reflect the real system as close as possible. The components used in the model are listed below and are explained in the following paragraphs. The total flow of the interacting terminal components is shown in Appendix 11.

Components	Role in the model
"Main"	Takes care of the simulation run & creates lay-out
"Generator"	Generates the vessels
"Import generator"	Generates import containers
"Export generator"	Generates export containers
"Berthmaster"	Manages the berth distribution
"Portainermaster"	Instructs the portainers
"rgv master"	Controls the number of Rgv's at the terminal
"Trafficmaster"	Creator of rgv routes and instructor of the Rgv's

The following components are "class components":

- Vessel
- Portainer
- Rgv
- Crane
- Stack
- Portloc (portainer location)
- Node
- Section
- Platform

In addition the model uses "macros" which are activated by a module to perform specific actions.

6.2.1 The component Main

Each PROSIM model requires the component Main. The Main takes care of the simulation run by creating components and their attributes. In this case the Main ignites a simulation run by activating the Shipgenerator. The Main also creates the portainers, the cranes, the rgv's, the stacks and the layout of the terminal. Figure 6.1 shows the different tasks of the Main. The shadow behind the first box refers to another process not shown in this figure, the creating of the lay out. This is explained in Paragraph 6.4.

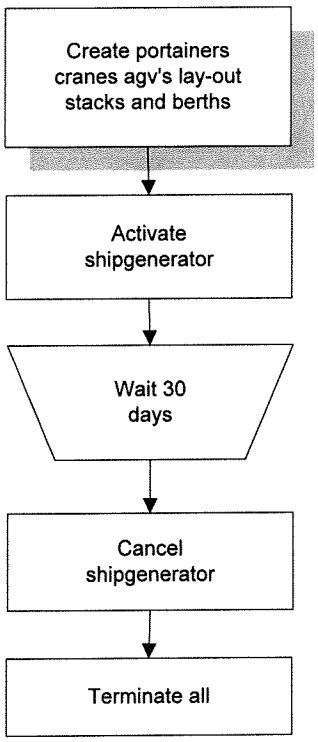


Figure 6.1, the process of Main

6.2.2 The component ship Generator

The component Generator is responsible for generating the vessels. Before generating a vessel it waits a certain inter-arrival time. This inter-arrival time is defined in the Main and refers to a distribution function. After a vessel is generated the Generator will again wait a certain period of time before creating the next vessel. This will go on forever until the Main stops the Generator. The process of the Generator is shown in Figure 6.2.

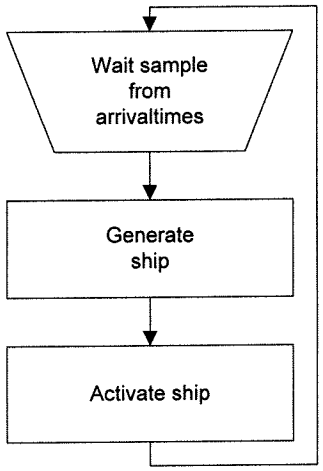


Figure 6.2, process of the ship Generator

6.2.3 Import generator & Export generator

These generators are responsible for creating the containers that will be imported to and exported from the terminal by the vessels. There are two different containers in this model: 20 foot and 40 foot containers. Over a longer period of time the Import generator has to create as many import containers as the Export generator creates export containers. The reason for this is the fact that this is a transhipment terminal and there is no hinterland connection.

6.2.4 The component Berthmaster

The Berthmaster is responsible for assigning the berths to the vessels, and is activated when a vessel is created or when a vessel is leaving the terminal. Once the Berthmaster is activated it looks whether there is a vessel waiting in the anchorage. If there is a vessel waiting in the anchorage the Berthmaster looks whether there is a free berth for this particular vessel. If there is a free berth the Berthmaster will allocate the free berth to this vessel, ask the vessel to moor and inform the Portainermaster of the new vessel at the berth. If there is no free berth the Berthmaster will instruct the vessel to wait in the anchorage.

In the model there are two kinds of berths, one berth for the mega vessels and two for the feeder vessels. If a feeder vessel arrives at the anchorage the Berthmaster first looks whether there is a free feeder berth. If there is only one free feeder berth the feeder vessel will be assigned to that particular berth. When there are two feeder berths available the Berthmaster will assign the feeder vessel to the berth that has not been occupied for the longest time. This way any possible shuffling at the other feeder berth can continue. The mega vessels have just one berth, Berth 2. Once a mega vessel enters the anchorage the Berthmaster looks whether the berth is free, independent of possible shuffling. If the berth is free the mega vessel will be assigned to the berth.

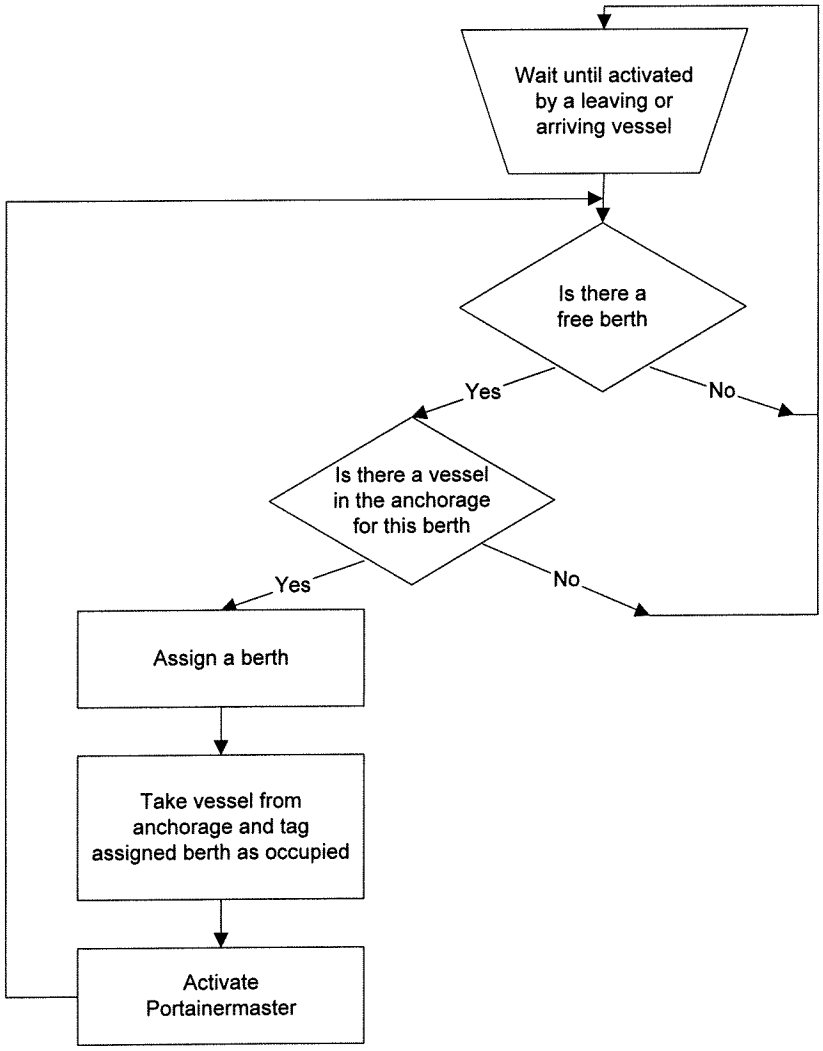


Figure 6.3, Process of the Berthmaster

6.2.5 The component Portainermaster

The model has been designed to contain two feeder berths and one mega vessel berth hosting a total of fourteen locations for portainers. It is possible to handle a mega container vessel with a maximum of six portainers and a feeder vessel with a maximum of four portainers. All the simulation runs in this study have been carried out with the maximum number of 14 portainers at the quay. This is done because the focus of this study lies on the processes on the terminal as opposed to a simulation of the arrival schemes of the container vessels.

It is possible to simulate with less than 14 portainers on the terminal if this is desirable. The main job of the Portainermaster will then be to allocate the portainers to the right location. An option could be to simulate a terminal with only 10 portainers available. This case has not been used in this study but the associated allocation will be discussed below.

Option: One vessel berthed

In case there is only one vessel berthed at the quay the Portainermaster assigns a maximum of 6 portainers to a mega container vessel moored at Berth 2 and a maximum of 4 portainers when it is a feeder vessel. The Portainermaster directs the portainers to the precise portainer locations on the quay. These locations depend on which berth the vessel is moored at.

Case: A mega container vessel is berthed at Berth 2:

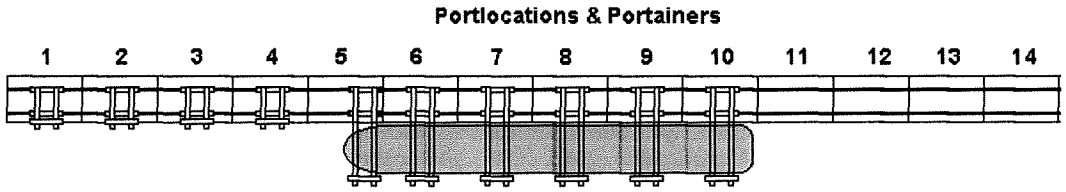


Figure 6.4, mega vessel moored at Berth 2

- The mega vessel moored at Berth 2 gets six portainers assigned on the portainer locations 5 to 10. The four remaining portainers will remain inactive on Berth 1.

Case: A feeder vessel is berthed at Berth 1:

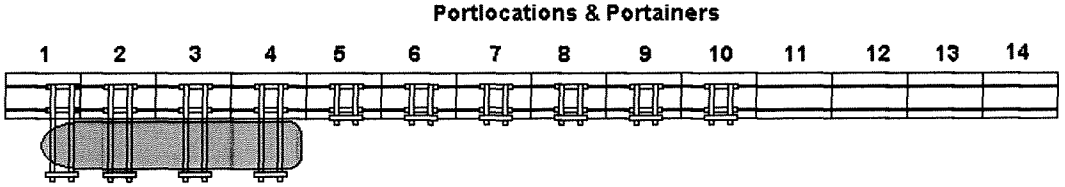


Figure 6.5, feeder vessel moord at Berth 1

- If moored at Berth 1, portainers 1 to 4 are directed to portainer locations number 1 to 4. The portainers 5 to 10 will remain inactive on Berth 2.

Case: A feeder vessel is berthed at Berth 3:

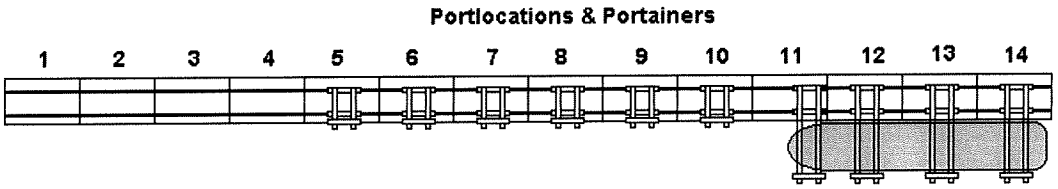


Figure 6.6, a feeder moored at Berth 3

- If the feeder vessel is moored at Berth 3, portainers 7 to 10 are directed to portainer locations 11 to 14. Portainers 1 to 6 will remain inactive on Berth 2.

Option: Two vessels are berthed

When there are two vessels moored at the quay a number of possibilities can be distinguished. There can be two feeders or a feeder and a mega container vessel.

Case: There are two feeder vessels berthed of which one is moored at Berth 1 and the other is moored at Berth 3.

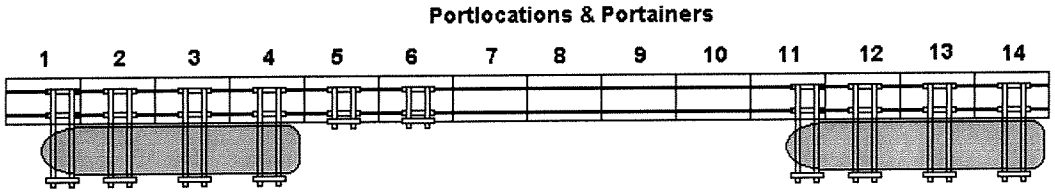


Figure 6.7, feeder vessels moored at Berth 1 & Berth 3

- The feeder vessel moored at Berth 1 will have the portainers 1 to 4 assigned to it, located on the portainer locations 1 to 4. The feeder berthed at Berth 3 will have the portainers 7 to 10 assigned to it, located on portainer locations 11 to 14.

Case: There is one feeder vessel moored at Berth 1 and one mega vessel moored at Berth 2.

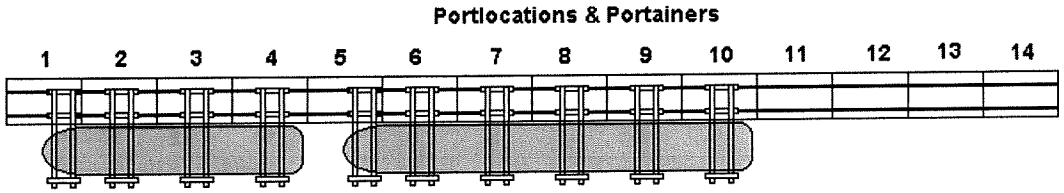


Figure 6.8, feeder vessel moored at Berth 1 and a mega vessel moored at Berth 2.

If the feeder vessel is moored at Berth 1 it will receive the portainers 1 to 4 located on portainer locations 1 to 4. The mega vessel will receive portainers 5 to 10 located on portainer locations 5 to 10.

Case: There is one feeder vessel moored at Berth 3 and one mega vessel moored at Berth 2.

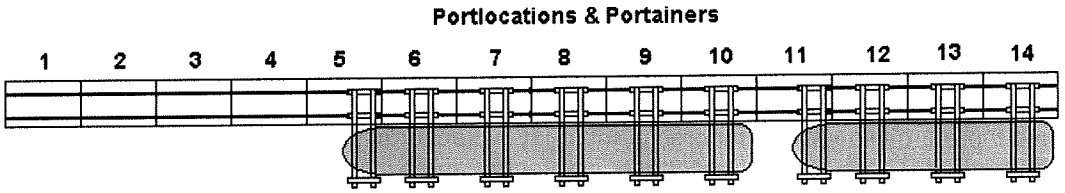


Figure 6.9, feeder vessel moored at Berth 3 and a mega vessel moored at Berth 2

- If the feeder vessel is berthed at Berth 3 it will receive portainers 7 to 10 on the portainer locations 11 to 14. The mega vessel will receive portainers 1 to 6 on the portainer locations 6 to 10.

Option: Three vessels berthed

When there are three vessels berthed at the quay, the only possibility is that there is a mega vessel moored at Berth 2 and that the two feeder vessels are moored at Berth 1 and Berth 3.

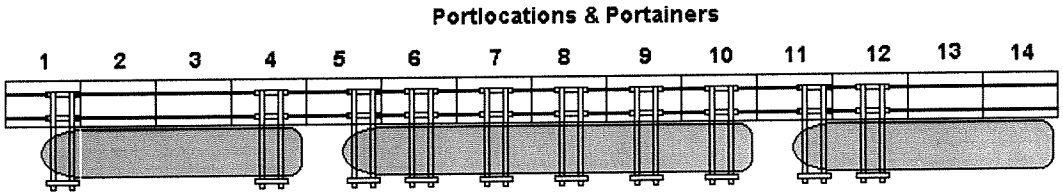


Figure 6.10, Three vessels moored

The mega container vessel receives six portainers on portainer locations 5 to 10, while the feeder vessels receive 2 portainers each. The portainers on Berth 1 and Berth 3 have variable portainer locations as they handle the entire feeder vessel. This has been done in order to serve the mega vessel as best as possible.

The consequence of working with 10 portainers is the occasional transferring of the portainers to the right portainer locations. The only situation in which the productivity on the berth is

seriously affected is when the terminal is fully occupied. Ultimately there will be an economic optimum for the number of portainers used on the terminal. As in this study the focus will lye on the processes that take place on the terminal, as opposed to a simulation study on vessel arrival schemes, the simulation runs have been carried out with the maximum of 14 portainers.

6.2.6 The component Rgvmaster

The Berthmaster allocates a berth to a vessel, and activates the Portainermaster. The Portainermaster in its turn allocates and activates the portainers. Once they have arrived on their allocated location, the portainers will activate the Rgvmaster. This is shown in Figure 6.10. Once the Rgvmaster is activated the Rgvmaster determines the number of active portainers and subsequently, the number of needed rgv's. The number of needed rgv's depends on the assignments of the active portainers at that moment.

If a portainer is unloading a vessel only 6 rgv's per portainer will be needed. If a portainer is fast-loading a vessel, the to be loaded containers will all come from the stack-cluster behind the portainer. In this case also 6 rgv's per portainer will be needed. When a portainer is adjacent loading a vessel, the to be loaded containers will have to come from the stacks located at an adjacent berth. In this case 10 rgv's per portainer will be used. These rgv's are all portainer dedicated. When a portainer is not yet at its right portainer location then no rgv's will be assigned to it.

This situation might occur when the terminal works with less than 14 portainers and when a portainer is not at his allocated location at the moment the Rgvmaster is activated. If a portainer has just finished loading a vessel and the vessel is about to leave, the shuffling of the unloaded containers to the correct yard can start.

Once the Rgvmaster has determined the needed number of rgv's and the number of already active rgv's he can calculate the difference. If there are not enough rgv's active, the Rgv-master will activate extra rgv's from the parking.

If there are too many rgv's the Rgvmaster will give the surplus of rgv's the attribute "bye". This attribute indicates that this rgv has to go to the parking. The next time this rgv asks the Trafficmaster for an assignment, the Trafficmaster will notice the attribute and the next assignment will be to go to the parking. The Trafficmaster will be explained in paragraph 6.2.7. This attribute is necessary because the Rgvmaster itself cannot send a rgv to the parking. The reason for this is that the rgv could be in the middle of an assignment, which it has to complete first. The process of the Rgvmaster is shown in Figure 6.11.

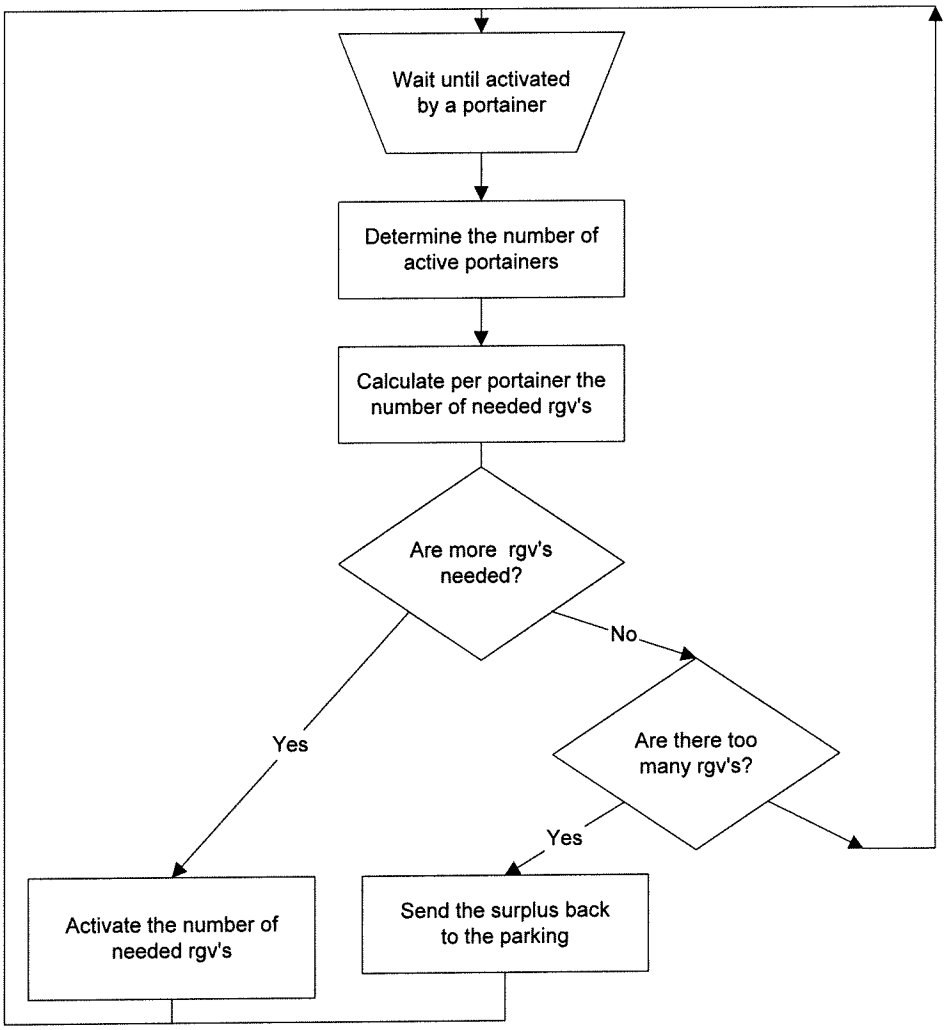


Figure 6.11, process of the Rgvmaster

6.2.7 The component Trafficmaster

The Trafficmaster is responsible for the assignments of the rgv's and can only be activated by a rgv. Once the Trafficmaster is activated, there are a number of possible assignments for the rgv's but roughly the assignments are divided into two main groups. The first group contains assignments concerning the horizontal transportation between the stack and the berth. The second group assignments concerns the shuffling between the stacks. The Trafficmaster can, by using the attributes of the rgv, determine the state of the rgv, which will lead to the next task of the rgv.

When activated by a rgv the Trafficmaster first looks whether the attribute "bye" of the rgv is true. If it is, the rgv is supposed to go to the parking. The Trafficmaster will remove the rgv from the "active rgv-list", a set containing all the active rgv's, and will activate the rgv from to_parking in the rgv module.

If the rgv is not supposed to go to the parking the Trafficmaster will, based on the state of the rgv's attributes: "yard", "loaded", "y" and "x" of the rgv, decide what the next the next assignment should be. These different assignments require different actions by the Trafficmaster. These actions are all stated in the macros of the Trafficmaster. Figure 6.12 shows the flow chart of the Trafficmaster.

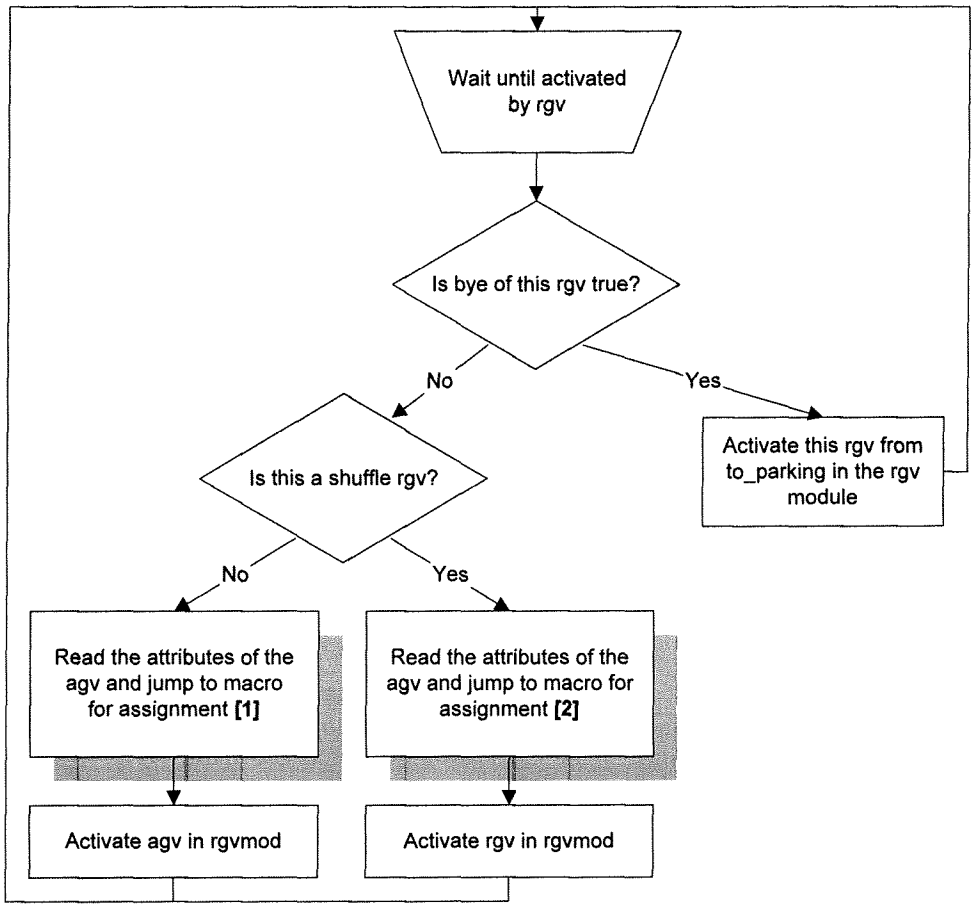
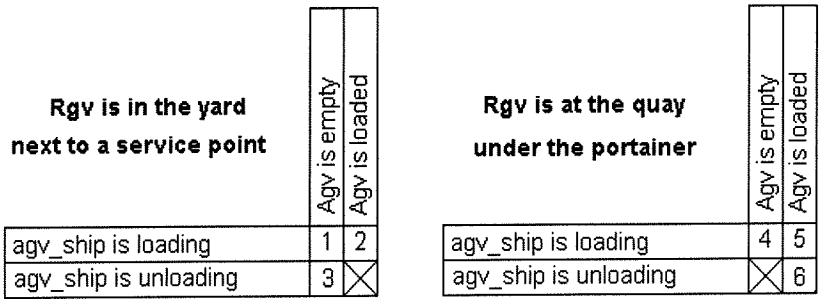


Figure 6.12, process of the Trafficmaster

In Figure 6.12, **[1]** and **[2]** refer to processes, depending on the attributes of the rgv. These processes will be explained below.

In case **[1]** the rgv is assigned to a portainer. The rgv's next assignment from the Trafficmaster depends on the location of the rgv and its attributes. The 6 possibilities are listed below. The rgv can be in the yard next to a service point where it has just delivered or received a container or the rgv is located under a portainer where it has just received or delivered a container.



Rgv is in the yard next to a service point	Agv is empty	Agv is loaded
agv_ship is loading	1	2
agv_ship is unloading	3	X

Rgv is at the quay under the portainer	Agv is empty	Agv is loaded
agv_ship is loading	4	5
agv_ship is unloading	X	6

Figure 6.13, possibilities when a rgv is working for a vessel

The numbers in the matrix refer to the actions of the Trafficmaster.

1. The Trafficmaster will first jump to a macro "loadyardallocation". In this macro, a stack is chosen for the rgv to retrieve a container by using an algorithm. This stack is now an attribute of this rgv. This attribute is called "y". Next the Trafficmaster will jump to the macro "rgv_stack_stack". This macro will create the "route" to this new stack. The Trafficmaster will also put the rgv on the assigned list of the stack. By the assigning of the rgv, the warehouse crane in the stack knows there is rgv on its way and can start retrieving the container to put it on a service point.

2. While the rgv is portainer dedicated the destination of the rgv is clear. The Trafficmaster will only jump to the macro "rgv_stack_berth" to create the route to get from the stack to the portainer location.

3. The Trafficmaster will jump to the macro "rgv_unload_back". This macro creates the route to get back from the yard to the portainer. The Trafficmaster will also assign the rgv to the portainer. When the rgv is assigned the portainer knows the rgv is coming and can start retrieving a container from the vessel.

4. The Trafficmaster will jump to the macro "loadyardallocation" and then it will jump to the macro "rgv_berth_stack". "Loadyardallocation" chooses the stack and "rgv_berth_stack" creates the route to get to this stack. The Trafficmaster will also put the rgv on the assigned list of the stack. By the assigning of the rgv, the warehouse crane in the stack knows there is rgv on its way and can start retrieving the container to put it on a service point.

5. This situation does not occur too often. It only occurs when a rgv has received the last import-container from a vessel and asks the Trafficmaster for an assignment. In this case the Trafficmaster will jump to the macro "unloadyardallocation". Here an algorithm decides where the rgv has to deliver the container. This stack is now an attribute of the rgv. Next the Trafficmaster jumps to the macro "rgv_unload", where the "route" of the rgv is created.

6. The Trafficmaster will jump to the macro "unloadyardallocation". In this macro the stack is chosen where the rgv has to deliver the unloaded container. Next the Trafficmaster will jump to the macro "rgv_unload", where the "route" to the stack is created.

In case **[2]**, the rgv is not attached to a portainer handling a vessel but is shuffling between the stacks located in the different yards on the terminal. During the shuffling process a rgv has been detached from its portainer. Instead of returning to its former portainer, it is now subscribed to the stack-cluster located behind the portainer. The shuffle containers, retrieved from stack cluster, will be delivered to one of the stacks located in the adjacent yard. If there is a container to be shuffled back the rgv retrieves it and brings it back one of the stacks located in its stack-cluster.
There are two states in which a shuffling rgv can activate the Trafficmaster in its shuffling section. Below these 2 possibilities are explained in succession.

1. The rgv is not loaded and is located under the portainer for the last time.
This is the case when an empty rgv has just delivered its last container to the portainer loading a vessel. Every rgv experiences this state only once during the handling of a vessel. In this case the Trafficmaster will first jump to the macro "yardshuffleallocation". In this macro a stack, located in the stack-cluster behind its former portainer, is found. From this stack the rgv will retrieve a shuffle container. This stack becomes an attribute of this rgv. Next, the Trafficmaster will jump to the macro "rgv_berth_stack". This macro creates the route from the berth to the stack.
The Trafficmaster will also assign the rgv to the stack. By the assigning of the rgv, the warehouse crane in the stack knows there is rgv on its way and can start retrieving the container to put it on a service point.

2. The rgv has just received a shuffle container and has to deliver it to another stack.
The Trafficmaster will first jump to the macro "yardsfuffleallocation". In this macro the stack, located at an adjacent yard, is chosen. The rgv has to deliver the container to this new stack. Next the Trafficmaster will jump to the macro "rgv_stack_stack". In this macro the "route" to the new stack is created.
The Trafficmaster will also assign the rgv to the stack. By the assigning of the rgv, the warehouse crane in the stack knows there is rgv on its way. If there is a shuffle container in the new stack destined to be taken away, the warehouse crane can start retrieving the container and put it on a service point.

Another important task of the Trafficmaster is keeping count of the number of containers that have to be loaded, unloaded and shuffled. Each vessel that arrives at the berth has an attribute “total unload”, referring to the number of containers to be unloaded. Whenever the Trafficmaster gives a rgv an assignment to get a container from the vessel it deducts one from the “total unload”. When the “total unload” of a vessel becomes zero, the Trafficmaster removes the vessel from the set “unloadingvessel” and joins it to the set “loadingvessel”.

When the vessel is being loaded the Trafficmaster also keeps count of the containers. The macro “container-yardallocation” is used for this. The number of loaded containers is called “tmtotal”. Tm stands for Trafficmaster. When “tmtotal” is zero, the vessel is fully loaded again and ready to leave the terminal.

6.2.8 The class component Vessel

There are two kinds of vessels, the mega container vessels and the smaller feeder vessels. After a vessel has been generated, it will activate the Import & Export generator that will create the containers for the vessel. The vessel will then enter the anchorage and activate the Berthmaster, while waiting for permission to berth. Once permission is granted the vessel will berth and wait while it is unloaded and loaded again. When the vessel is loaded it will leave the berth and activate the Berthmaster. After it has left the berth the attributes of this vessel will be recorded and the vessel will be terminated.

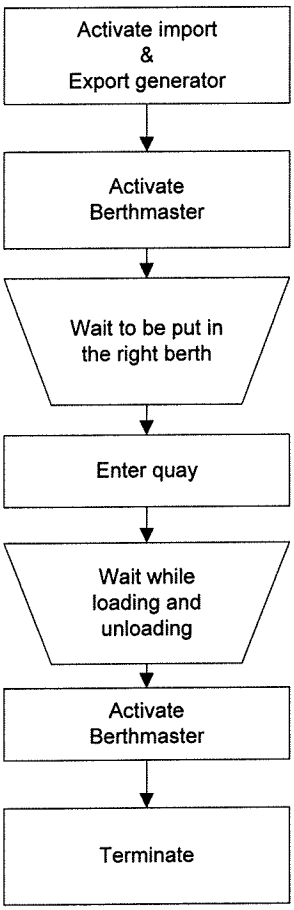


Figure 6.14, process of the class component vessel

6.2.9 The class component Portainer

Portainers are designed for the transport of containers between the quay and the vessels. The Portainermaster assigns the portainers to the vessels and sends them to the right locations on the quay. These portainer locations in the simulation program are called "portlocs". In the simulation runs 14 portainers are used for 14 portainer locations. This means the portainers will always be on the right location. It is possible to work with less portainers in the program.

The portainer can be activated by the Portainermaster or by an assigned rgv. If it is activated it checks whether it is located at the right portainer location. If not, it will try to get to the right location this only happens when less than 14 portainers are used. It may happen that a portainer, that is not at its the right location on the quay, has to wait before it is allowed to move because other portainers that are still working, are standing in the way. These portainers are supposed to clear the way as soon as possible (such portainers may be in the process of unloading or loading which has to be finished first).

Once the portainer has arrived at the correct location it will wait for a rgv to be joined to the assigned list of the portainer. If this assigned rgv is an empty rgv the portainer will get a container from the vessel and wait for the rgv to arrive under the portainer, in the portainer queue. When the rgv arrives, the portainer will put the container on the rgv. Once the unloading of the vessel is finished, the portainer will start the loading of the vessel at the arrival of the first loaded rgv under the portainer.

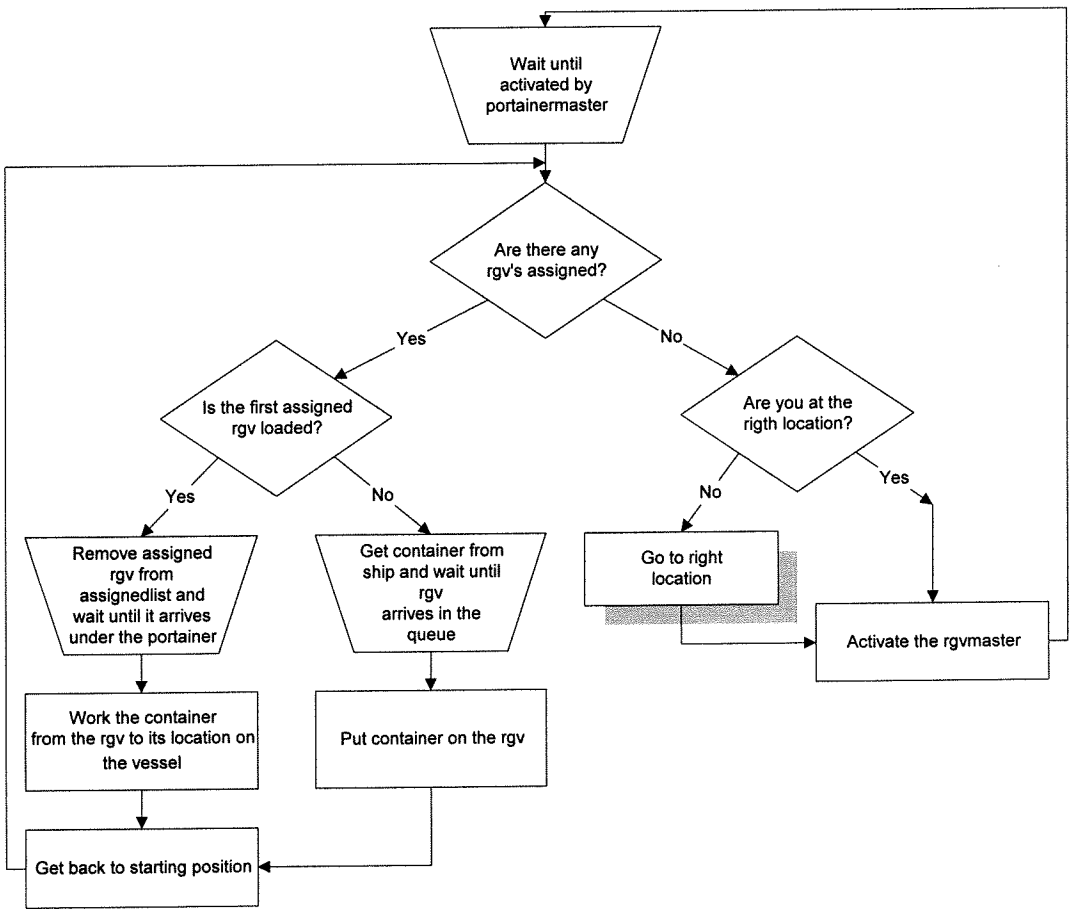


Figure 6.15, process of the class component portainer

The grey area in the flow chart in Figure 6.15 indicates this box is part of another not shown process, it is the process where the portainer transfers to its right location.

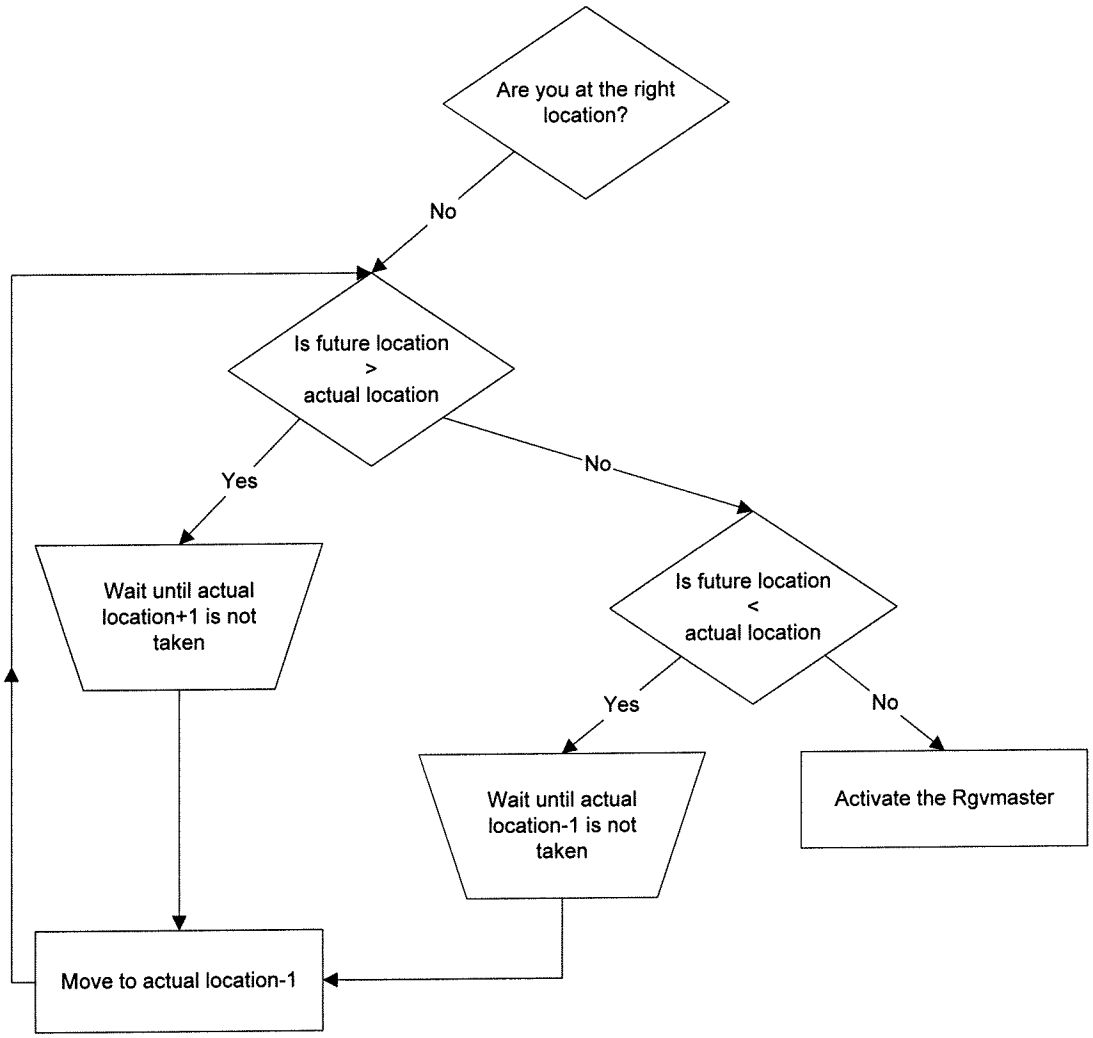


Figure 6.16, flow chart of the portainer while it is going to the right position at the quay

6.2.10 The class component Rgv

Rgv's are used for the horizontal transportation between the quay and the stack. One rgv is assumed to carry a maximum of one container; this can either be a twenty-foot or a forty-foot container. The Rgvmaster activates the rgv's. Once a vessel is moored at the quay and the portainers are activated the Rgvmaster decides, using the number of active portainers, how many rgv's to retrieve from the parking.
The Rgvmaster joins all the needed rgv's to an "active rgv" list and to an "rgv worklist". The "active rgv list" is a set containing all the active rgv's. The rgv worklist is a set containing all the rgv's requesting the Trafficmaster for work. Once joined to the rgv worklist, the rgv activates the Trafficmaster. The Trafficmaster removes the rgv from the rgv worklist, decides where the rgv has to go and activates the rgv in the rgv module. The flow chard concerning the Trafficmaster shows that there are some different possible options from where to activate the rgv.

Option: The Trafficmaster activates the rgv from "toyard" in the rgv module

Toyard means that the rgv is not in the yard but at the quay and is destined to go to the yard. This as opposed to "inyard" where the word "in" indicates that the rgv is already in the yard and has to go to another place in the yard, for example during shuffling.
In this case the rgv is activated in the rgv module from toyard, so the rgv jumps to the macro called "rgv_berth_stack". When the rgv enters the macro its present and the future location on the terminal are known. The macro creates the route, which the rgv has to follow in order to get to the future location, using algorithms. In this case the route is from a berth to a stack. The rgv drives using its route, to the right location and enters the warehouse crane queue to and waits to be serviced.

Option: The Trafficmaster activates the rgv from "inyard" in the rgv module.

If the rgv will be activated in the rgv module from "inyard", the rgv jumps to a macro called "rgv_stack_stack" in the Trafficmaster. When the rgv enters the macro its present and future location at the terminal are known. The macro creates the route, which the rgv has to follow in order to get to the future location, using algorithms. In this case the route is from a stack to a stack. The rgv drives using its route, to the right location, enters the crane queue and waits to be serviced.

Other options are that the Trafficmaster activates the rgv from "toquay", "inquay" and "toportainer". The macro's respectively jumped to by the rgv to receive the routes are "rgv_stack_berth", "rgv_unload_back", "rgv_unload" and "Rgv_toparking".

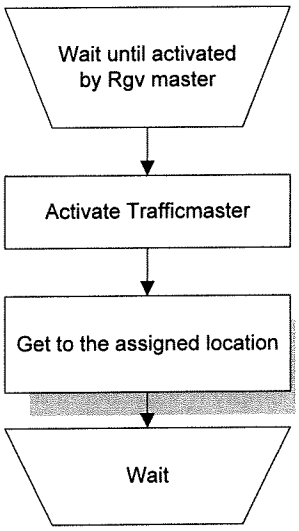


Figure 6.17, process of the rgv

The grey parts in Figure 6.17 contain the driving processes of the rgv. This driving process is shown in chapter 6.5.

6.2.11 The class component Crane

There is one warehouse crane on every pair of stacks. A pair of stacks contains two stacks, two warehouse lanes and two rows of service points. The warehouse crane is situated in between these two service point rows, as can be seen in Chapter 5.
The warehouse crane is responsible for the horizontal and vertical transport from the service points to the racks of the warehouse and back. The warehouse crane will not be active until a rgv is assigned to the warehouse crane or until a rgv has delivered a container onto a service point.

The Trafficmaster can assign a rgv to a warehouse crane. When for example the Trafficmaster has just ordered an empty rgv to collect a container from a stack, the Trafficmaster will join the rgv to the assigned list of the warehouse crane working on that stack. The moment the rgv is on the assigned list the crane becomes active. The warehouse crane checks the list and learns there is a rgv coming with a container request. The crane starts to retrieve the container from the stack and loads it onto a service point. When subsequently the rgv arrives in the stack the container is already on the service point and ready to be taken away.

Another possibility is that a rgv arrives in a stack to deliver a container. In this case the rgv is not on the assigned list of the warehouse crane, because the warehouse crane cannot work in advance. When this rgv arrives at the stack a service point shuttle will put its container on a service point. The warehouse crane will notice there is a container on the service point and will deliver it to the right place in the stack.

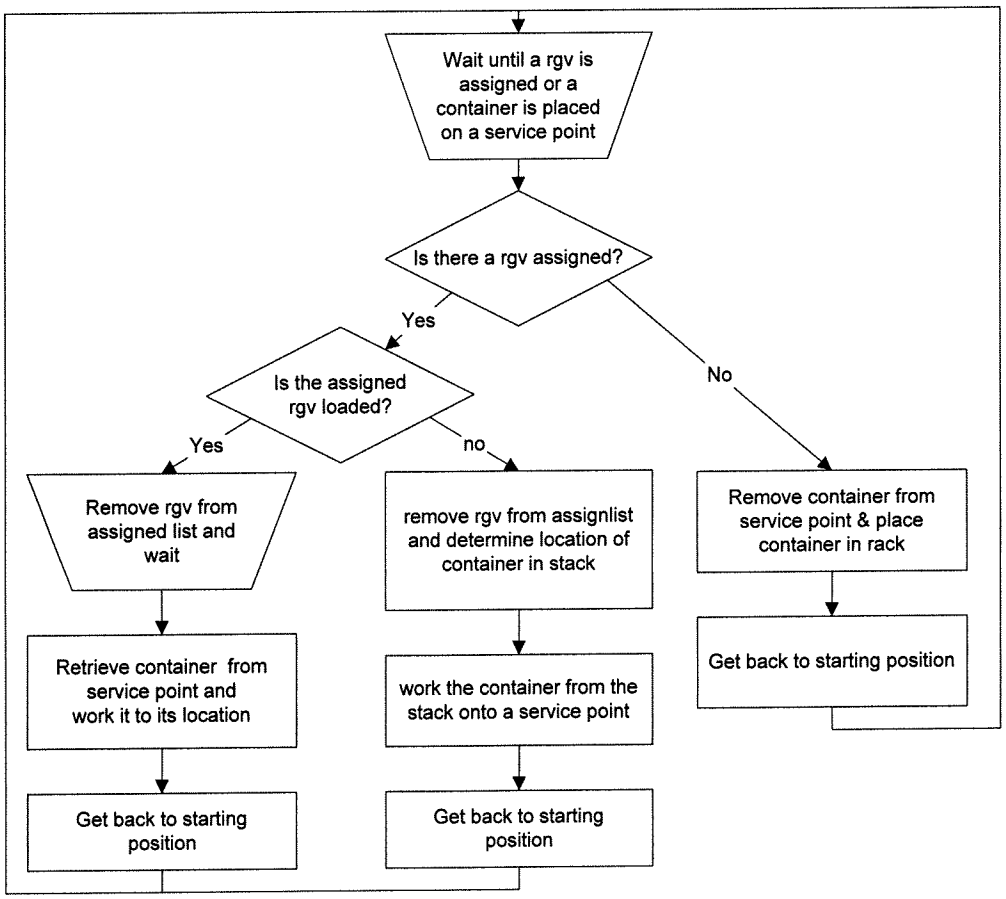


Figure 6.18, process of the class component crane

6.3 The macros used in the program

In addition to the modules the program uses macros. Macros are used to perform specific actions. The different macros used will be explained here. The listings of the macros are shown in Appendix 28, the listing of the program.

- Create platform, this macro is responsible for creating the platforms and is used by the Main. This is explained in paragraph 6.4.
- Station, this is a sidetrack and is used by a rgv when it leaves the main track in order to be served by a portainer.
- Short cut, this is a sidetrack used when a rgv wants to leave the main track to take a shortcut, or enter a stack.
- Stacking crane is a macro used by a rgv leaving the main track in order to enter a warehouse lane to be served by a warehouse crane.
- Rgv_to_parking is a macro used by the rgv in order to get to the parking
- Rgv unload is a macro that creates the route for a rgv to get from the berth to a stack. This macro is used by the Trafficmaster to determine the route for the rgv when the rgv is unloading a vessel.
- Rgv_unload_back is a macro used by the Trafficmaster to create the route for a rgv when unloading a vessel. This route is from a stack to a berth.
- Rgv_stack_berth is a macro used by the Trafficmaster that creates a route from any stack to any berth.
- Rgv_berth_stack is a macro used by the Trafficmaster to create a route from any berth to any stack.
- Rgv_stack_stack is a macro the Trafficmaster uses to create the routes between the stacks.
- Create_stack is used by the Main when it creates the layout of the terminal. In this macro the stacks are created.
- Super_shortcut is a sidetrack used by a rgv when it wants to enter a shortcut at a place at the terminal connected to two or more sidetracks.
- Unloadyardallocation is a macro that determines the stack locations for the rgv's to deliver their containers to.
- Loadyardallocation is a macro that chooses where the rgv has to get a load container.
- Conversation is the macro that creates the control box.
- Preclaimnodes is a macro called upon by the rgv responsible for claiming nodes when more than one node needs to be claimed.
- Rgv_uneasy_b_s this is a macro that creates a route from a berth to a stack in the transition between loading a vessel and shuffling at a portainerlocation.
- Yardshuffleallocation is the macro that creates the locations for the rgv's to get containers from when shuffling.
- Y_creation is a macro responsible for determining which stacking crane works in which stack.
- Create_outportation is the macro responsible for creating outportations.
- Crane_cycletime is a triangular function that can be used when creating crane cycle times.
- Crane_outportations is the macro that creates output files for all the cranes used at the terminal. With these files the mean cranecycletimes can be calculated.
- Creating sections is a macro that attaches existing queues of sections to windows in order to monitor these queues. In this way node rows and stack rows can be monitored.
- Create_portout is a macro that creates portainer outportations in order to monitor the queues at the portainers.
- Portainer times is a macro responsible for creating the portainer cycletimes here four functions are used. Two for the loading and unloading of the mega vessels. And two for the loading and unloading of the feeder vessels.

6.4 The layout of the terminal as created in the Main

Appendix 2 shows the layout of the terminal as it is created in the Main. To understand the creation of the terminal layout some used components must be explained. These are the class components node, section, stack, portlocation and platform.

The class component node

The structure of the terminal consists of nodes. Nodes are points in the 2-dimensional layout area of the terminal. By giving these nodes different attributes driving lanes and sections can be created. Important attributes of a node are: claimed, node speed, side_track and outs[1], outs[2] and outs[3]. These attributes will be explained later on the basis of some examples. The 800 nodes and their attributes are created in the Main module.

The class component section

This class component section is a so-called dead class component. Dead class components differ from other class components in the fact that they do not have their own module in the simulation program. These components do not do anything in the system but the other components can do something with them. Dead class components are only used to carry information.

Sections can be one way driving lanes created between nodes. These one way driving lanes are called the "main tracks" of the terminal. These main tracks are created in the Main through an input file. In Figure 6.19 a part of this file is given as an example; the complete input file is shown in Appendix 16.

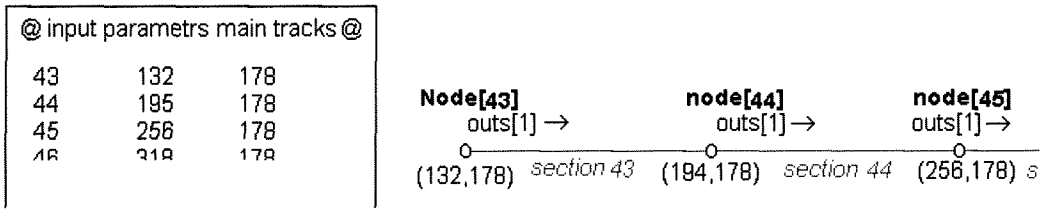


Figure 6.19, the creation of a main track by the Main

When the Main creates the layout of the terminal, it first to reads the first character string of the input file. In this case the number "43". This number refers to a node, node[43]. The second and third character strings are the x and y co-ordinates of this node, "8" and "178". After reading a second node and its attributes the main creates a section between node[43] and node[44]. This section is called section 43. An important attribute of this section is its length called "s_length". The function "distance_to" calculates this s_length. After creating section 43 node[43] receives a new attribute "outs[1]". Outs[1] refers to section 43. The endpoint of section 43 is node[44].

The class component stack

Stacks are created in the main using a macro and an input file. Some important attributes of the stacks are a stackrow, a stacknumber and a way in and out of the stacks. Whenever a rgv enters a stack it can do so by driving to the stack_in_node. This is a reference to a node. Via the way_in_stack, a reference to a section, the rgv enters the stack.

By entering the stackrow the rgv can be served. In the end the rgv will leave the stack using the way_out_stack. In Figure 6.20 the file for the creation of stacks is given.

```
@ warehouse lanes@
stack21  231  260  163  146
stack22  259  232  163  134
stack23  233  258  163  131
stack24  257  234  163  119
stack25  235  256  163  116
```

Figure 6.20, input file for creating warehouse lanes

The Main reads the first character string from the file. This is the name of the stack, "stack21". The second character string is the entrance node of the stack. In this case node[231]. The third character refers to the endpoint of the stack, node[260]. The fourth and the fifth character strings refer to the actual x and y co-ordinates of the middle of the stack. The main creates the stack by creating 2 sections a section "way-in" and a section "way-out" of the stack.

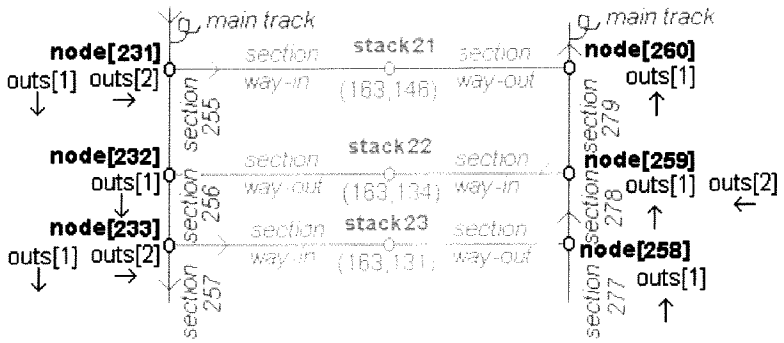


Figure 6.21, the creation of a stack

After creating the stack, the entrance node of the stack, in the case of "stack21" node[231],will receive an attribute outs[2]. Outs[2] is a reference to the section "way-in". Figure 6.21 shows the creation of the stack.

A rgv can thus enter stack21 by leaving the main route at node[231]. It subsequently enters the section way-in. The warehouse lanes are the blue lanes. The arrows point out the driving directions.

Shortcuts

Shortcuts are lanes that can be created between two nodes that are not connected by a main track. Shortcuts can be used as a connection between two main routes. A rgv can enter a shortcut like it enters a stack, via a node. Shortcuts are created using an input file. An example of a part of this input file is given in Figure 6.22.

```
@short cuts@

75      44
45      74
```

Figure 6.22, input file for creating shortcuts

The Main reads the first character string, in this case 38. This refers to node[38], the entrance of the short cut. The second character string of a node refers to the end point of the shortcut. The shortcut itself is the section between these nodes. After the main has created the shortcut the entrance node of the short cut will receive an attribute outs[2]. An example of a shortcut is given in Figure 6.23.

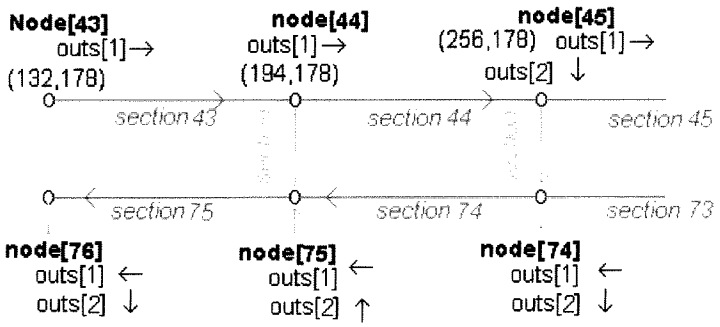


Figure 6.23, the creation of a shortcut

The shortcuts are the green lanes between the red main tracks.

The class component portloc

The class component portloc is created in the Main. A portloc is a place where a portainer can be situated, a portainer location. There can be more portlocs than portainers so it is possible for a portloc to be unoccupied. This can be the case when a simulation is carried out with less than 14 portainers.
Every portloc has a set called "portlocqueue". This set contains the rgv's waiting at that portlocation to be served. The lengths of the portlocqueues during a simulation are recorded and can be watched during or after the run.
Another attribute of a portloc is the set "assigned rgv's.

This set contains rgv's performing a task for the portainer located at that portloc. When a rgv is assigned, the portainer knows it cannot move to another portloc. This is to avoid a rgv driving to an unoccupied portloc.

The class component Platform

This class component platform is created in the Main by jumping to the macro "create_portainerlocation". A portainerlocation consists of two platforms. This has everything to do with the layout of the terminal. These platforms are also created via an input file. An example of this file input is given in Figure 6.24.

```
@portainers@

loc3los     242   249   163   58
loc3laad    248   243   163   50
```

Figure 6.24, input file for creating a platform

The Main reads the first character string, in this case "loc3los". This is the name of the platform. The second and the third character strings refer to the entrance and the exit node. The fourth and the fifth character strings refer to the x and y co-ordinates of the actual location of the platform. In Figure 6.25 the creation of the platform is given.

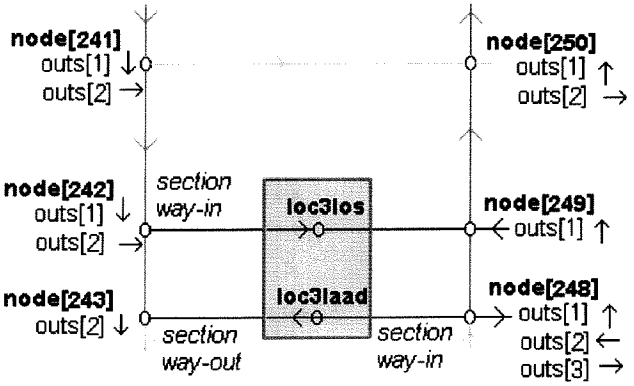


Figure 6.25, the creation of 2 platforms and a portainer location

The blue area in Figure 6.25 is called a portainer location. The black lines are the platforms. Each portainer location has two platforms. Meaning it has two ways for a rgv to enter the portainer location. In the end they will both end up in the same "portlocqueue". Loc3los and Loc3laad are the names of the platform.

6.5 *The traffic rules at the terminal*

6.5.1 The priority rules

The layout of the terminal is divided in main tracks, short cuts, platforms and stacks as has been explained in the previous section. Once a rgv is on a main track, the red lanes in Figures 6.19, 6.21, 6.23 and 6.25, it has priority over rgv's entering from a sidetrack. Sidetracks can be short cuts, platforms or stacks.

When driving on the main track the rgv automatically claims the next node that enters the claiming zone of the vehicle and declaims this node when it has past the node. Another rgv can only enter the main track when the entrance node onto the main track is not claimed. So when a rgv has been served at a stack and wants to enter the main track it has to wait until the entrance point, the endpoint of the stack, is not claimed. This implies that a rgv on the main track only stops when it has to leave the main track.

Another rgv driving behind this rgv will always leave enough space for the first rgv to brake, to turn its wheels and leave the main track. This space is called the claiming zone of a rgv. As a consequence of the priority rule, there will be no queuing on the main track, but it is possible a rgv has to wait when it wants to enter the main track coming from a side track. This is shown in Figure 6.26.

At all the entrance points of the main tracks room must therefore be reserved for the waiting rgv's that are not yet permitted to enter the main track. This waiting area is available in the green section.

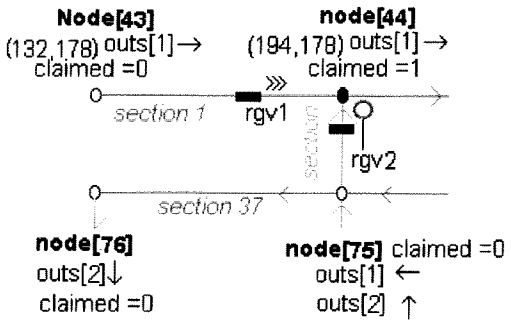


Figure 6.26, the claiming of nodes when driving on the main track

In Figure 6.26 Rgv 1 is driving on the main track. During this driving on the main track Rgv 1 automatically claims the nodes within claiming distance. Once a node is claimed its attribute "claimed" becomes 1. Rgv 2 coming from a shortcut wants to enter the main track at node 44 but has to wait until "claimed" of node 44 becomes 0 again. "Claimed" of node 44 can only become 0 again when Rgv 1 has passed the node and changes the attribute "claimed" of the node to 0.

While Rgv 2 is not permitted to enter the main track it has to wait in the queue before node 44. The lengths of these queues are recorded by the simulation program and can be analysed during or after the simulation run. The green sections also enable a rgv to leave the main track without hindering the traffic at the main track.

An enlargement of the area around node 44 is given in Figure 6.27.

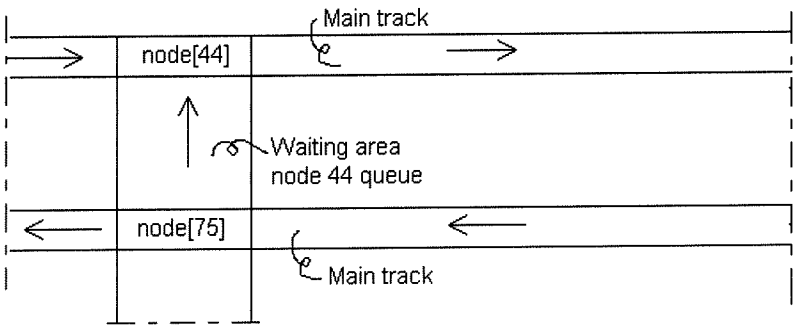


Figure 6.27, space for waiting queue at node 44

6.5.2 The driving of a rgv

A rgv drives over the terminal using its own attributes and the attributes of the nodes on the terminal. Two of the most important attributes of the rgv are the set "route" and the set "entrance". The route of the rgv is a set containing all the required exit nodes from a main track. Entrance is a set of the rgv containing only the starting point of the journey. This is a reference to a node. The Trafficmaster creates these sets before a rgv starts its journey. The most important attributes of the nodes are the "outs[1]", the “outs[2]”, the "outs[3]" and the "claimed" of the nodes.

The rgv starts its journey at its entrance node. This can be the exit of a stack or the exit of a portainer location. When the rgv starts the entrance node becomes the current node although the rgv has not yet entered the main route. The rgv can only enter the main route when the current node is not claimed. If the current node is not claimed, the rgv enters the main track and claims the current node and starts driving.

Once the rgv enters the main track the rgv will automatically follow the main track driving direction and claim all the nodes within claiming distance. This process continues until the rgv arrives at the first node of its set "route". This node is called the exit node. When the rgv arrives at the exit node the rgv will leave the main track and enter a short cut, a portainer queue or a stack queue.

If the rgv enters a stack or a portainer the rgv will enter the waiting queue and will wait to be served. If the rgv enters a shortcut it will pass the shortcut and will try to re-enter the next main track. This process continues until the rgv reaches its destination. The driving process of the rgv is shown in Figure 6.28. This process always starts at a portainer location or in a stack.

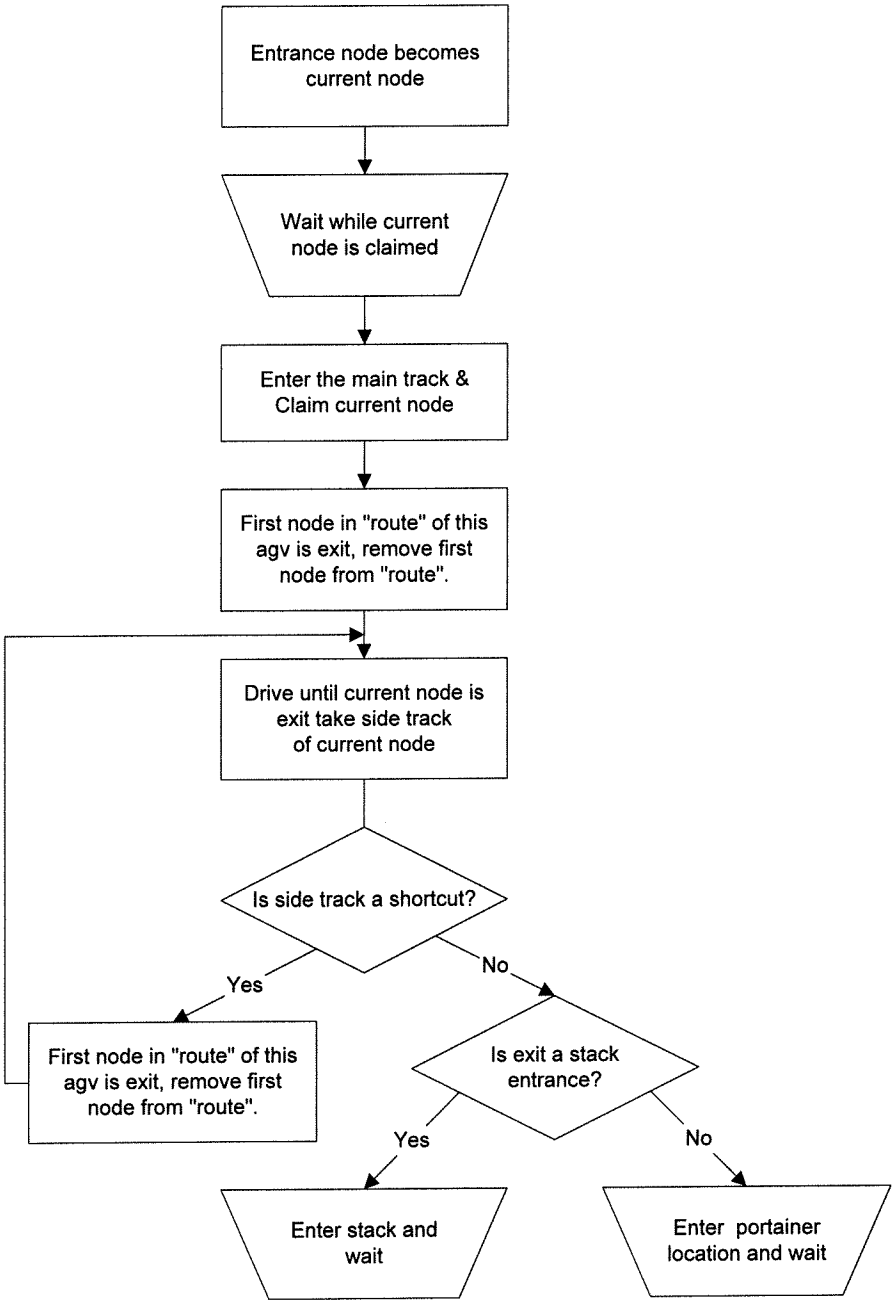


Figure 6.28, flow chart of the driving process of a rgv

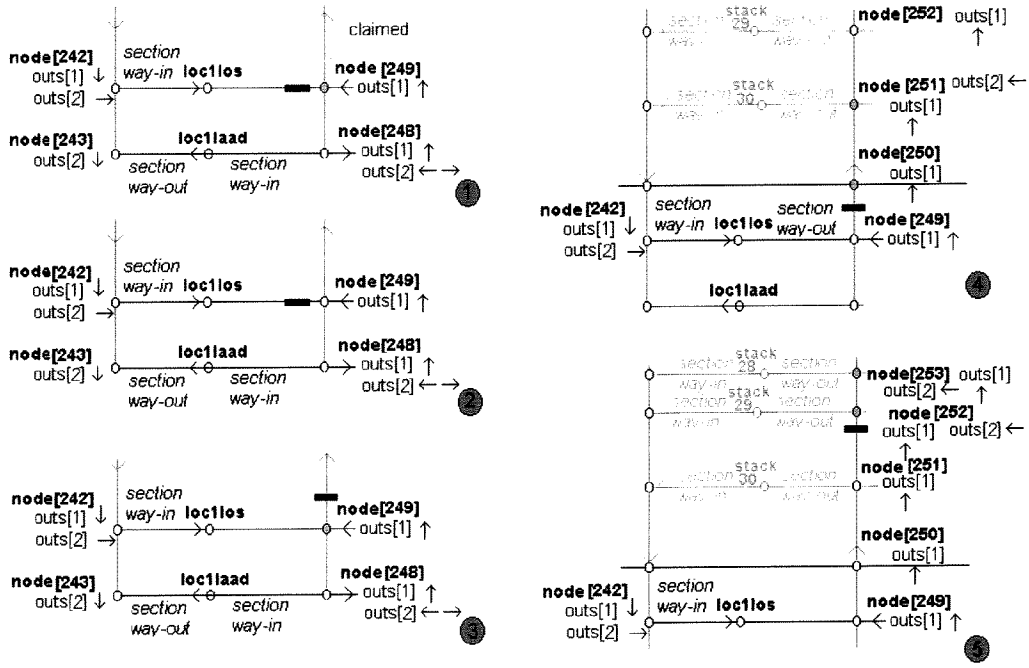


Figure 6.29, the driving of the rgv

In Figure 6.29 the flow chart process of the rgv is applied to a situation on the terminal. In this example the rgv has to go from a portainer location to stack 28. The set "route" of the rgv contains just one node, node 253. The set "entrance" contains one node, node 249.

1. The entrance node, node 249 becomes the current node of the rgv. The rgv has to wait while the current node is claimed by another rgv.
2. When current node is not claimed anymore the rgv is permitted to enter the main track.
3. The rgv enters the main track and claims all the nodes within claiming distance.
4. The rgv starts driving on the main track, automatically claiming all the nodes within the claiming distance of the rgv. By claiming node 250 and 251 the rgv makes it impossible for rgv's coming from stack 29 or stack 40 to enter the main track.
5. After the rgv has passed node 250 and 251 it declaims these nodes. The rgv will continue driving until it reaches node 253, the exit node. At this node the rgv will leave the main track and enter stack 28.

7 Verification of the simulation program

Normally when a simulation program has been written the process of checking whether the program is correct, consists of two phases. The first phase is the verification process in which is checked if the simulation program acts like it is supposed to do. The second phase is the validation phase where the simulation program is compared to results gained by experience. In this study only a verification study has been made. The main reason is that there is no experience with this type of terminal, which is necessary for the validation process.

Before any reliable results can be obtained from simulation runs, the program must be checked for inaccuracies and errors that may influence the results. Inaccuracies might be present in the computer simulation language or in the design of the program. The compiler easily traces inaccuracies in the computer language. The compiler is a function that indicates all the errors in the different modules and macros.

Inaccuracies in the program itself are more difficult to trace. They can be traced in two different ways; by stepping through the program and checking the state analyses of the components or by simulating with different extreme values for the input parameters and observe whether the results are as expected.

In the first paragraph the algorithms that determine the destinations and the routes of travelling rgv's have been inspected. Next, in the second paragraph, the travelling rgv itself will be followed closely. In Paragraph 8.3 the consequences of a blocked node are illustrated. The influence of the increase of the number of rgv's per portainer on the rgv cycle times are dealt with in Paragraph 8.4, followed by an impression of the portainer handling function. Finally the influence of the number of rgv's on the ship dwell times at the berth has been verified.

7.1 Verification of algorithms

During a simulation rgv's are send to different areas on the terminal depending on their tasks. To determine the destination of a rgv, the program uses so called allocation macros. If, for instance, a rgv has just received a container from a portainer it will call on an allocation macro in order to receive a stack number.

Since the rgv's are always dedicated to one particular portainer it is not necessary to receive a portainer number after each assignment. The portainers can be seen as a base from which the agv's operate from during the unloading or the loading of a vessel. As a result the allocation algorithms only provide the agv's with a stack number.

After the allocation macros have determined the destination of the rgv, the rgv will also receive a route in order to get there. This route is a collection of exit nodes. When the rgv arrives at such an exit node the program identifies the node as an exit and instructs the rgv to leave the main track it is on. The macros that provide these points are the route macros.

7.1.1 Verification of the allocation algorithms

In the allocation macros "unload-yard-allocation", "load-yard-allocation" and "yard-shuffle-allocation" rgv's receive their new destinations in the yard. During the unloading of a vessel, a just loaded rgv, located under its portainer, will call upon the "unload-yard-allocation". This macro will use its algorithms to determine where the rgv has to deliver the newly acquired container.

After the vessel is unloaded the rgv's will retrieve the containers from the stack so that the vessel can be loaded. Once a rgv has delivered a container to its portainer, it will call upon the loadyardallocation in order to receive a stack number from where it will retrieve the next container.

The rgv's will start shuffling when the last container has been loaded. During shuffling the rgv calls upon the yardshuffleallocation every time it needs a new destination in the stacks.
It is important to know that the algorithms used in the allocation macros are correct, in the way that the distributed allocations are random and evenly divided.

The three allocation algorithms have been monitored by attaching outstrings to their results. These outstrings contain the "Y" values. A "y" value represents the destination of the rgv. If, for instance, y is 1 it means the destination of the rgv is warehouse lane 1, located in Yard 1. These values are registered and converted to an output file. In this file the stacks or warehouse lanes and the number of rgv entries have been printed. This has been plotted in a histogram as can be seen in Figure 7.1. If one stack is visited considerably less or more than other stacks this can be easily spotted.

Allocation of stacks during unloading/fast-loading of a vessel

Rgv's unloading a vessel deliver the containers to the available stacks located behind the portainer they are working for. The rgv receives the stack number of the destination stack from the macro "unloadyardallocation". This unloading run has been performed with a mega vessel moored at Berth 2. Six portainers have been used in the unloading process.

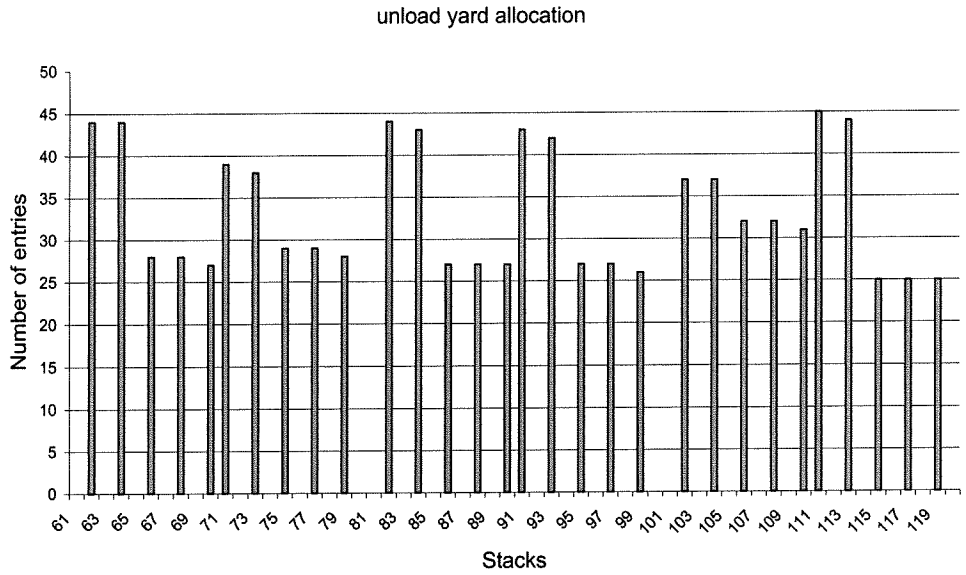


Figure 7.1, allocation of stacks in Yard 2 during the unloading of a mega vessel

Figure 7.1 shows a relative higher concentration of rgv entries in the stacks containing the 20' containers, these are the stacks 61-64, 71-74, ... , 111-114. This trend is the result of the equal amount of 20' and 40' containers in the yard due to the assumed TEU factor of 1.5. Because the ratio 20' stacks to 40' stacks is 2:3; the ratio of rgv entries in the 20' stacks and 40' stacks also equals 2:3.

Another important issue is that during the unloading of a vessel half of the warehouse lanes available behind the portainer are used. This is caused by the orientation of the warehouse lanes in the stacks. Warehouse lanes 61 to 70 are located behind portainer 5, but only the even warehouse lanes are used. These are the ones orientated in the direction of the unloading cycle. This does not mean that the stacks located on top of the uneven warehouse-lanes are not used while unloading. The warehouse crane is located between an odd and an even numbered warehouse lane and can deliver or retrieve a container from all the racks. As explained in Chapter 4, Figure 4.12.

Allocation of stacks during the adjacent loading of a vessel

Figure 7.2is the result of a run with a single feeder vessel moored at Berth 1 during adjacent loading. The consequence is that the stacks located in Yard 2 behind Berth 2 supply the containers for the feeder vessel moored at Berth 1. As can be seen in Figure 7.2, these are the stacks 61 to 120. The stacks located behind Berth 1 and Berth 3 are not involved in the loading procedure.

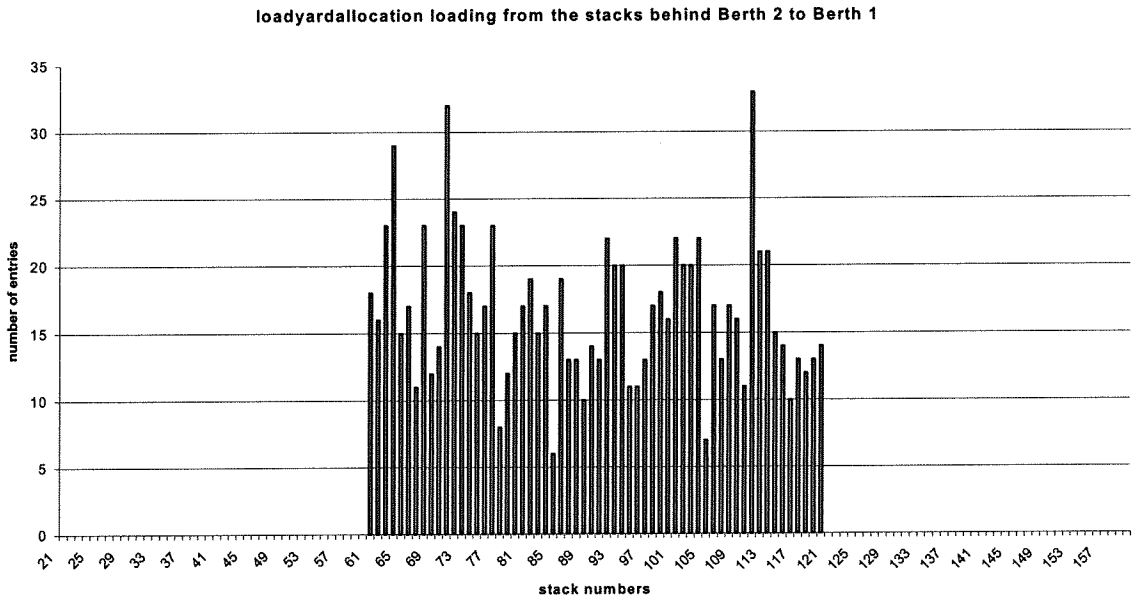


Figure 7.2, allocation of the stacks during the adjacent loading of a feeder vessel moored at Berth 1

As is the case during unloading, the 20' container stacks have a higher average concentration of agv entries. During adjacent loading all the stacks are involved as opposed to the unloading/fast-loading of a vessel.

Allocation of stacks during shuffling

For the verification of the "yard-shuffle-allocation", a shuffle procedure has been simulated. After a feeder vessel, moored at Berth 1, has been unloaded and loaded, the containers unloaded in Yard 1 will be transported to Yard 2. During a shuffle cycle a rgv makes two allocation requests. One to retrieve a shuffle container from a stack located in Yard 1 and one to deliver the concerning container to a stack located in Yard 2. The stacks from which the containers are to be retrieved are the stacks 1-60. The destination stacks are the stacks 61-120 located behind Berth 2 in Yard 2. The stacks 121-180 in Yard 3 behind Berth 3 are not involved in this run.

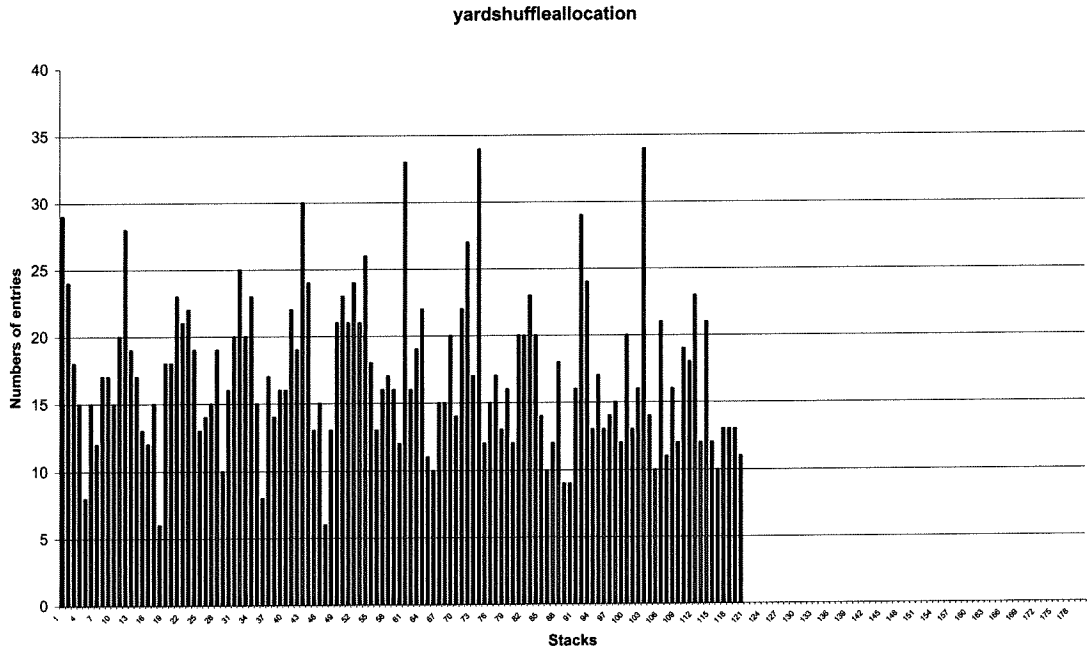


Figure 7.3, rgv entries in the stacks during a shuffle process

The stacks holding the 20' containers are, again, visited more frequently, which complies with the expectations.

7.1.2 Verification of route generating algorithms

The macro's agv-stack-stack, agv-berth-stack, agv-stack-berth, agv-unloading and agv-unloading-back are used to create the routes of the rgv's to the desired locations like allocated stacks or portainers. As was stated in the introduction of this chapter, the routes of the rgv's consist of exit points. If a rgv arrives at such a point it will exit the track it is on. The route generating algorithms always generates the exit nodes that lead to the shortest route.

Possible errors in the algorithms can be traced by performing simulation runs were all the different route possibilities are created. When a rgv has an error in its route it means the rgv will never arrive at the allocated location. Instead it will generate an error or the rgv will be caught in a loop. In both cases the errors are easily observed.

7.2 Verification of the route of a travelling rgv

It is essential for the simulation that the rgv's move exactly according to the planned route and planned time scheme. By following all the movements a rgv carries out during the transportation of a container, the various stages that take place have been checked.
Only one rgv has been activated in order to keep the tracing simple. Acceleration, wheel turning and the loading and unloading of containers by the warehouse cranes and the portainers have been checked.
An explanation of the various driving situations that take place and the step by step trace can be found in Appendix 12.

7.3 Interaction between different rgv's at crossing nodes

When a rgv wants to enter a main track at a node that is already claimed by another rgv, it will have to wait until the latter has passed by and clears its claim on the node. To check whether a rgv is waiting while the node is blocked, some nodes at shortcuts have been artificially claimed during a run. The expected result is a jam behind the shortcut where the rgv's will wait in vain to enter a main track.

As an example Node 504 has been artificially blocked. Even rgv's have to cross Node 504 when they are providing the Portainers 1 and 2 with containers from Yard 2 during adjacent loading.

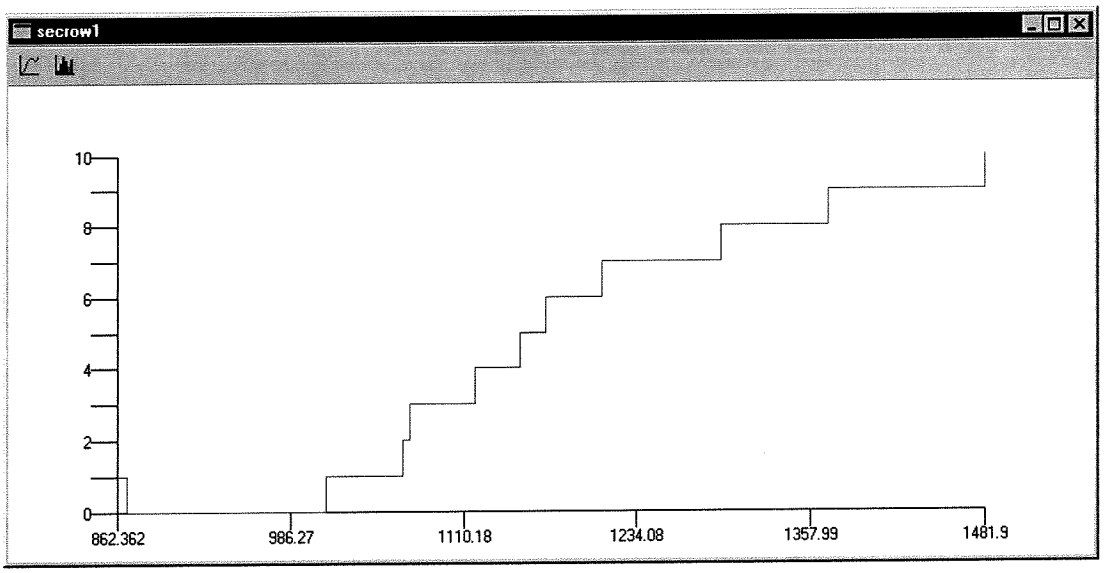


Figure 7.4, the queue behind a claimed node

In Figure 9.4 the horizontal axis projects the time, which is registered in seconds. The number of rgv's in the queue is registered on the vertical axis. After 986 seconds Node 504 was artificially blocked. This automatically prevents all the rgv's from crossing the node. A queue of waiting rgv's is the result.

7.4 A qualitative analysis of the rgv cycle times

Having verified the behaviour of the rgv on individual rgv level in Paragraph 7.2, it is also essential to analyse the rgv cycles for each of the 3 possible rgv cycles. These are the rgv cycles during the unloading of a vessel, the loading of a vessel and when shuffling containers from one stack to another. Increasing the number of rgv's per portainer will influence the rgv cycles.

By plotting the duration of the rgv cycles a qualitative interpretation can be made. The average time needed for a rgv to complete each of these cycles is the average rgv cycle time. In the following paragraphs each of the 3 rgv cycles will be further described and analysed.

The rgv cycle times during the unloading of a vessel.

During the unloading of a vessel the rgv cycle starts under the portainer where the rgv receives a container. The rgv will deliver the container to one of the stacks located behind the unloading portainer. After the delivery the rgv returns to the portainer to receive the next container.

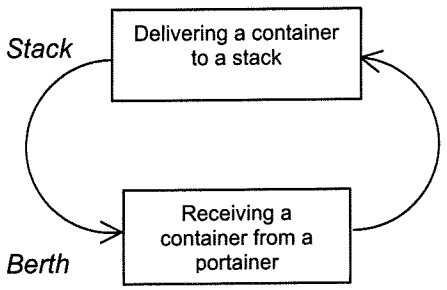


Figure 7.5, unloading cycle of a rgv

A rgv will unload its containers in the stacks located directly behind the portainer. This implies that an unloading rgv will be influenced during the transportation by rgv's that work for the same portainer and by rgv's that work for an adjacent active portainer. Using one active portainer with one rgv during the unloading of the vessel will eliminate these effects. This way an undisturbed average rgv-cycle time per portainer location can be determined.

The relative longer travelling distances to the stacks behind the even portainers cause the fluctuation in the series with one rgv, which can be seen in Figure 7.6. This as opposed to the shorter travelling distances to the stacks behind uneven portainer locations. The undisturbed average cycle time of one rgv on an uneven portainer location is 3,9 minutes while a rgv located on an even berth has an undisturbed average cycle time of 4.5 minutes.

If the numbers of rgv's per portainer are increased the expectation is that the average cycle times of the rgv's will also increase. More rgv's per portainer equals a higher chance of rgv's interfering with each other on crossings, resulting in waiting times.

The runs with more than one rgv have been carried out with 14 active portainers on the three berths, causing interactions between rgv's of adjacent portainers. If the number of rgv's is increased to 2 or even to 6 per portainer, the average rgv cycle time will indeed be longer as

can be seen in Figure 7.6. With 6 active rgv's the average cycle time for even portainers is about 5 minutes, while the average cycle time for uneven portainer is about 5.5 minutes.

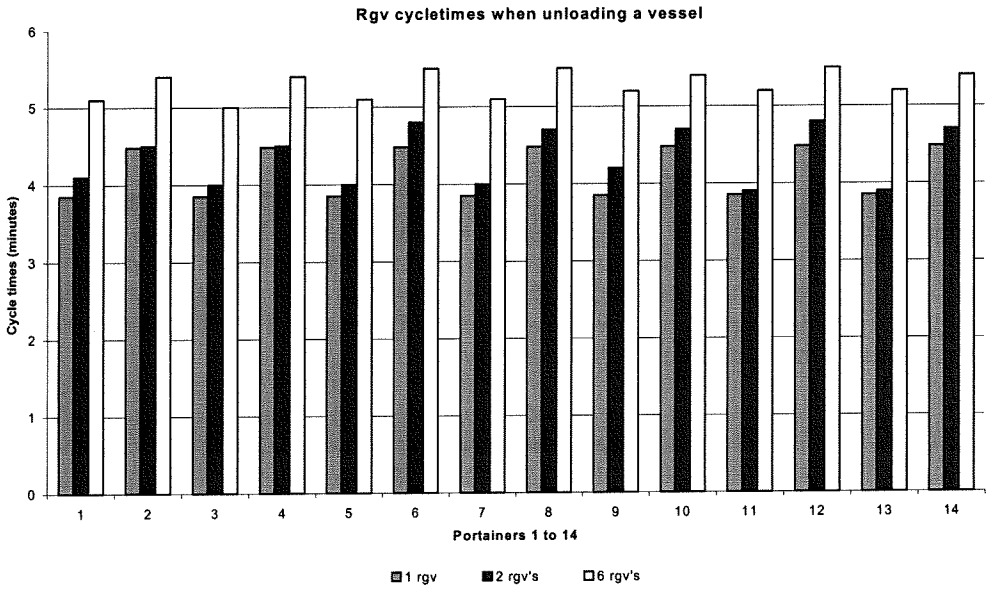


Figure 7.6, rgv unloading cycle times when unloading a vessel

The rgv cycle times during the adjacent loading of a vessel.

During the adjacent loading of a vessel the rgv cycle starts in the stack, where the rgv receives a container. The agv will deliver the container to its portainer located on one of the three berths. After the delivery the rgv returns to a stack to receive the next container.

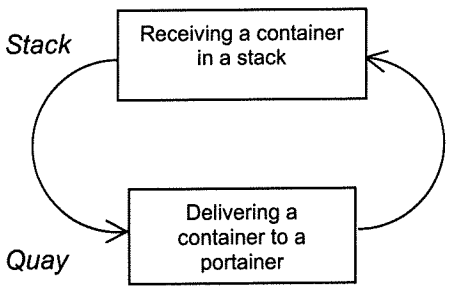


Figure 7.7, loading cycle of an rgv

A vessel can be loaded in two different ways. One method is by loading containers that have been shuffled to the correct stacks located behind the loading portainer. The routes of the loading rgv's are similar to routes unloading rgv's cover. Verification has demonstrated that the rgv cycle times during fast-loading are the same as during unloading. Therefore Figure 7.6 can also be interpreted for fast-loading.

When there has been insufficient time on the terminal to shuffle all the containers to the correct positions behind the loading portainers, the rgv's will have to retrieve the containers from an adjacent berth, as explained in Chapter 3.5. This means the rgv will have to cover a greater distance in order to collect a container resulting in a longer rgv cycle time.

During 'adjacent' loading the rgv's of the portainers on Berth 1 will retrieve their containers from the stacks situated in Yard 2 behind Berth 2. As can be seen in Figure 7.8, the rgv cycle times of the rgv's subscribed to the portainers on Berth 1 decrease from Portainer 1 to Portainer 4. This is plausible, as Portainer 1 is located further from Yard 2 than Portainer 4.

Rgv's subscribed to the portainers on Berth 3 will also retrieve their containers from Yard 2 resulting in the same rgv cycle times as the rgv cycle times of the rgv's subscribed to the portainers on Berth 1. Berth 2 is the exception because the rgv's will either retrieve their containers from Yard 1 or Yard 3 resulting in fairly invariable rgv cycle times.

This basic verification run has not been carried out with 1 agv per portainer but with 2 rgv's per portainer, because the rgv flow during adjacent loading is separated into two directions. Even rgv's take the bottom inter-berth lanes located on the berths whereas uneven rgv's take the top inter-berth lanes located behind the yards. A second run with 12 rgv's illustrates that by increasing the number of rgv's the rgv cycle times will be longer. This is plausible since more rgv's amount to more interactions between rgv's.

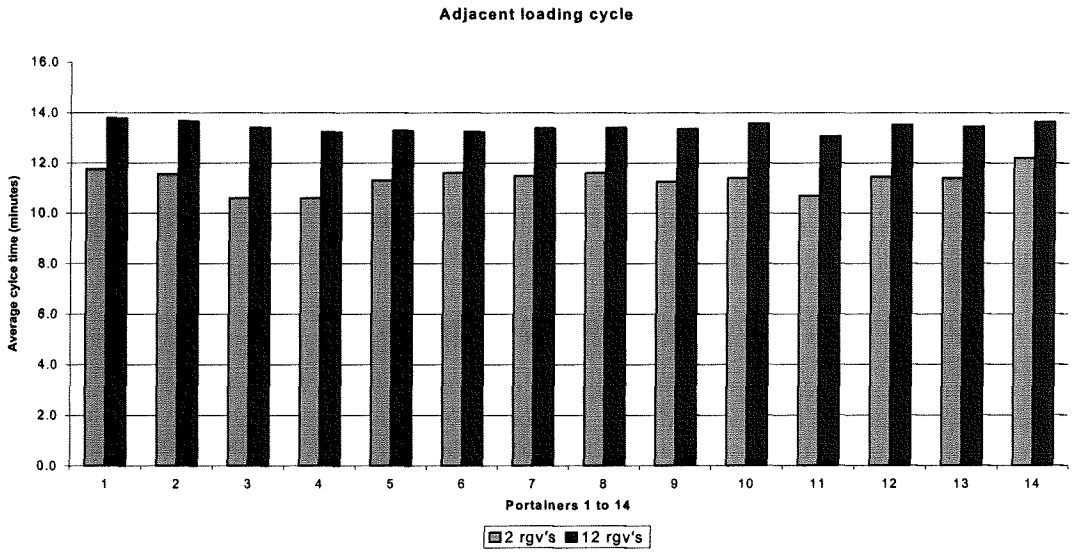


Figure 7.8, rgv cycle times when adjacent loading a vessel.

The rgv cycle times for shuffling on the terminal.

A shuffle cycle starts at the supplier of containers. In this case the supplier is one of the 10 stacks located behind the portainer the rgv was subscribed to during the unloading and the loading of the vessel. The rgv will always return to this cluster of stacks to retrieve a shuffle container. Stack-cluster 1 for instance is the 10 stacks located behind Portainer 1.

Once the rgv has received the container, it will head for a stack located behind the berth of destination. Rgv's with origin Berth 1 and Berth 3 will shuffle containers from Yard 1 and Yard 3 to Yard 2. Rgv's with origin Berth 2 will shuffle containers from Yard 2 to Yard 1 and Yard 3. After delivering the container there are two possibilities, depending on the fact whether there are containers to be shuffled back. Either the rgv will receive a new shuffle container and deliver it to the rgv's original stack-cluster, or the rgv will return empty to its original stack-cluster.

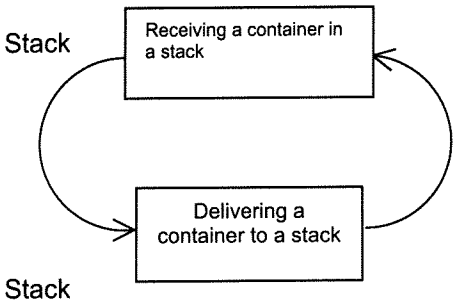


Figure 7.9, shuffling cycle of a rgv

When comparing Figure 7.8 and Figure 7.9 the similarity is noticeable. Like Figure 7.9 the two most remotely located stack-clusters have the longest rgv cycles. The rgv cycle times of the rgv's operating for the stack-clusters located behind Berth 1 and Berth 3 drop as the stack-clusters are situated closer to Berth 2. The rgv cycle times of the rgv's subscribed to the stack-clusters behind Berth 2 are more or less constant.

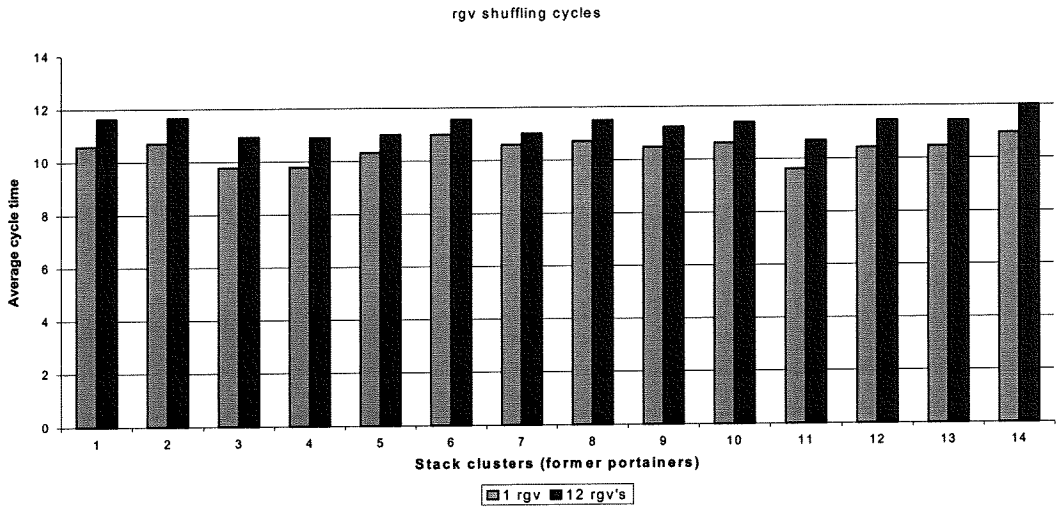


Figure 7.10, rgv cycle times during shuffling

The main difference between both graphs is that the rgv cycle times during shuffling are shorter. The reason for these shorter cycle times is that the rgv's working for the stack-cluster behind a portainer location do not depend on one portainer, as is the case when loading a vessel. When shuffling containers the rgv's are spread over the warehouse cranes operating in the stacks, thereby reducing the waiting queues.
The travelling distance between two stacks located in different yards is more or less the same as the distance between a stack and a portainer located on different berths.

7.5 Portainer handling function

The tooth-like portainer handling function can be verified by linking the working times of the portainers to a so-called pointstream, which is a form of output data. During the handling of a feeder vessel all the four portainers have been verified resulting in the graph printed below. This graph shows the following portainer cycle-times during the unloading and loading of a feeder vessel.

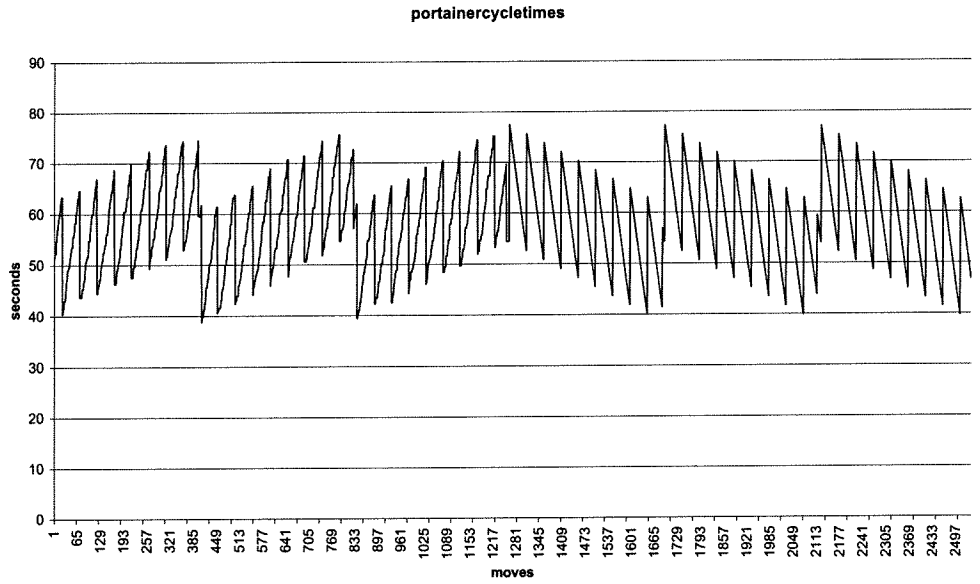


Figure 7.11, the portainer cycle times

In Figure 7.11 moves 1 to 1250 represent the unloading of the feeder, implying 1250 containers will have to be unloaded, this means each portainer will unload approximately 313 containers.

The unloading starts with the top container from the quayside of the vessel. The time needed to retrieve this container is relatively short, as can be seen in Figure 7.11. After the first container the portainer works horizontally towards the waterside of the vessel, taking away the top layer of containers. After this layer the portainer takes away the second layer. The process of taking away one layer creates one small tooth like shape in the portainer cycle times. In total there are 9 layers in this feeder vessel. The cycletimes of the successive layers become higher while the portainer has to go deeper inside the vessel. The three repetitions during the unloading procedure represent the same actions of the portainers after they have moved sideways on the quay to get to another hold.

The second part of Figure 7.11 moves 1267 to 2485 represent the loading part. The small teeth are reversed as a result of the fact that when the portainer starts loading, it starts with the container placed on a location deep in the vessel at the seaside of the feeder-vessel. This figure shows that the shortest cycle time is about 39 seconds and the longest about 77 seconds. The mean cycle time for the loading and unloading of the vessel is 58 seconds.

The portainer cycle times of a mega vessel are similar to the feeder handling times. The only difference is the number of smaller teeth. As a mega vessel is assumed to have 18 horizontal container layers instead of 9.

7.6 Influence on ship cycles when varying the number of rgv's

In this verification study a calculation is made on the number of needed rgv's for the loading and the unloading of a vessel. It is expected that when the number of rgv's per portainer is increased, the dwell times of the vessels at the terminal will become smaller. At a certain amount of rgv's per portainer the portainer will have a permanent waiting-queue. At this point adding more rgv's will only result in larger waiting queues saturating the terminal.

A variety of runs have been done in order to check the increase of portainer production, by increasing the number of rgv's per portainer. Figure 7.12 shows the production of four portainers unloading a feeder vessel moored at Berth 1. During all the runs the mean portainer cycle time was 58 seconds.

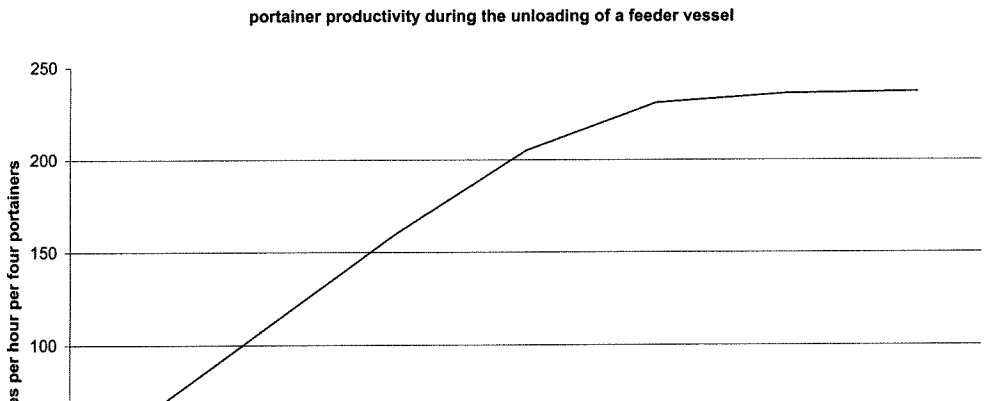


Figure 7.12 portainer productivity during the unloading of a feeder vessel

With one rgv per portainer, the total amount of active rgv's is four. As the rgv unloading cycle takes about 4 minutes, it is not surprising to see that the production of the four portainers is about 60 moves an hour. Increasing the number of rgv's per portainer will result in a higher portainer production.

If more than 6 rgv's per portainer are used to unload the feeder vessel, the portainer production will not increase any further. A surplus of rgv's will only increase the queue under the portainers.

Figure 7.13 shows the production of four portainers during the adjacent loading of a feeder vessel moored at Berth 1. The containers that are to be loaded are randomly divided over the stacks in Yard 2 situated behind Berth 2.

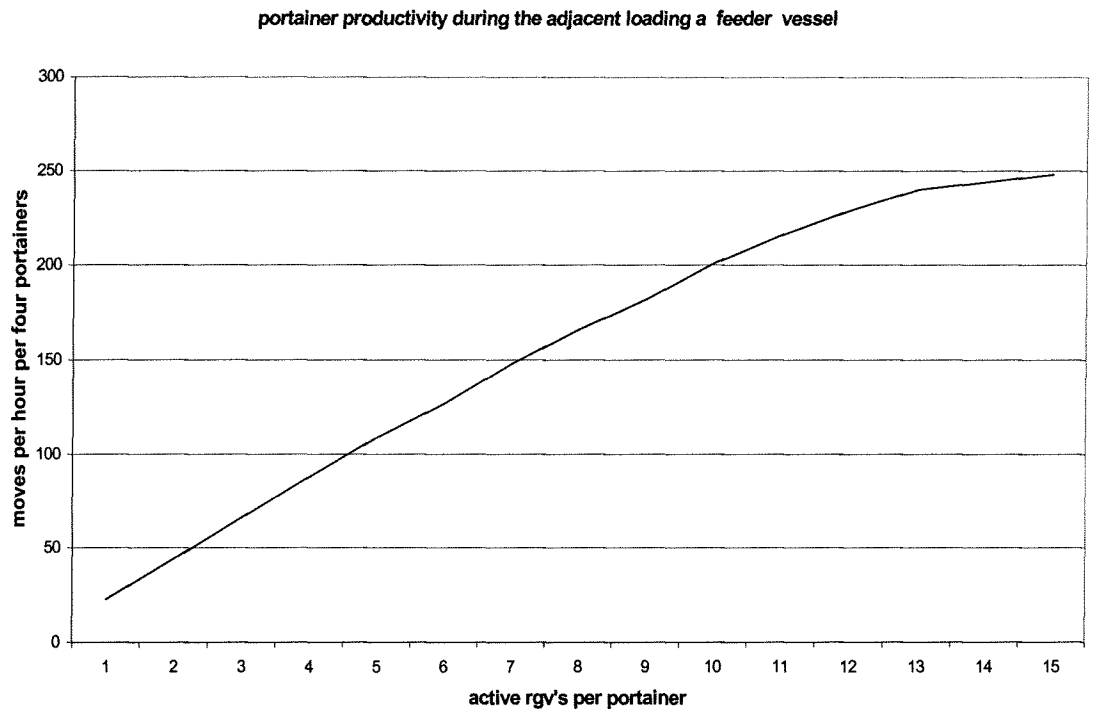


Figure 7.13, portainer productivity when adjacent loading using a different number of rgv's

Figure 7.13 shows the increasing portainer productivity versus the number of rgv's used per portainer. As the rgv cycles during adjacent loading are longer than during unloading, the production of the berth with 1 rgv per portainer is less than during the unloading of the feeder vessel. The berth production with one rgv per portainer is about 25 moves per hour.

When 15 rgv's are used the portainer production is almost 250 containers per four portainers per hour. With an average portainer cycle time of 58 seconds this implies the occupancy rate of the portainers is 100 percent, implying there is always a rgv waiting under the portainer to be served. The consequence of using this much rgv's is the increase of queues under the portainers as can be seen in Figure 7.14 where the x-axis represents the time in seconds.

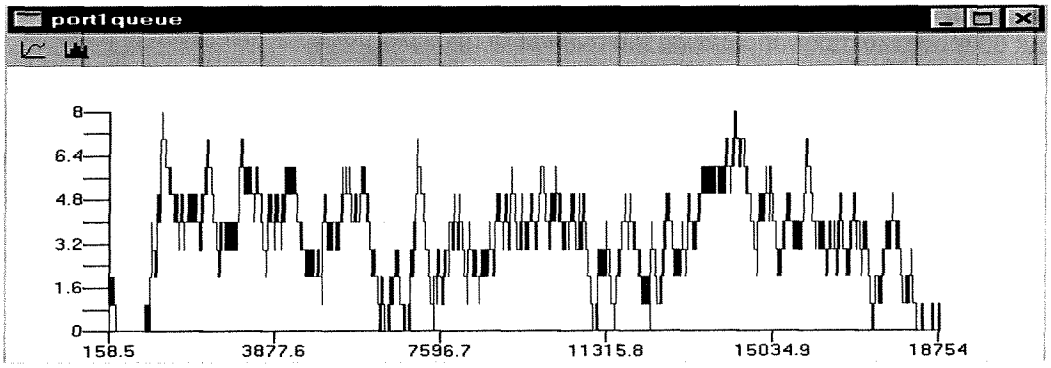


Figure 7.14, Portainer 1 queue using 15 rgv's per portainer during the adjacent loading of a feeder

8 Simulation and results

After having verified the reliability of the simulation program in Chapter 7, the results of the simulation runs will be analysed in this chapter. The basis of the simulation runs is the reference terminal as described in Chapter 8.1. The performance indicators will be described in Chapter 8.2. In Chapter 8.3 and 8.4 the reference terminal has been analysed during the handling of one vessel. By handling one vessel at a time the portainers, rgv's and warehouse cranes will not be interfered by rgv's working for other vessels. This way the maximum undisturbed terminal production for the concerning vessel can be determined.

Following these simulation runs the pressure on the terminal has been increased by simulating with three occupied berths in Chapter 8.5. By operating on more than one berth, the production during mutual interference can be determined. Simulation runs concerning two occupied berths have not been carried out. This is based on the assumption that if the achievements of the terminal with 3 occupied berths are sufficient, the achievement of the terminal with 2 occupied berths also is.

8.1 Determination and description of the reference terminal

Before any simulation runs have been made, a standard reference terminal was determined. This means that a standard number of rgv's, portainers and cranes have been chosen. The performances of these three terminal components have already been determined and illustrated in Chapter 4, but will be included for the sake of completeness. The same applies to the properties of the two types of vessels, as they have been illustrated in Chapter 3.2.
By determining a standard terminal and carrying out the simulation runs, the performances of the terminal can be calibrated, making it possible to analyse variations on the reference terminal in a later stage.

8.1.1 Standardisation of the container vessels

Standardisation of the feeder vessels and the mega vessels simplifies the model making it easier to analyse the performances of the three terminal components and the terminal as a whole.

Characteristics of the vessels
A standard feeder vessel in this simulation has a capacity of 1575 TEU and will be handled by four portainers. Each visit the feeder vessel will be completely unloaded and loaded again, bringing the total amount of handled containers to 3150 TEU. Applying a TEU ratio of 1.5, 2100 moves/call have to be carried out by the portainers during the unloading and the loading of the vessel.

A standard mega vessel has a capacity of 8000 TEU and will be handled by six portainers. The terminal will unload and load 4725 TEU being a 60 % share of the total capacity of the mega vessel. This brings the amount of handled TEU during a mega vessel visit to 9450 TEU. Applying the 1.5 TEU ratio means portainers will have to make 6300 moves per mega vessel visit.

8.1.2 The standardisation of the terminal components

The components on the terminal are the portainers, the horizontal transportation and the container storage system. In the following paragraphs the standardisation of the components on the terminal is dealt with.

The number of Portainers
The number of portainers on the quay during the basic runs is determined by the maximum occupation of the berths. When all three the berths are occupied, 14 portainers can be operational. Two feeder vessels moored at Berth 1 and Berth 3 demand four portainers each, while one mega vessel moored at Berth 2 demands 6 portainers.

Handling performance of the Portainers
The variable handling performance of the portainers that has been plotted in Figure 7.11 in Chapter 7.5 is the standard handling performance of the portainers.

The attributes of the Rgv's
The properties of the rgv, acceleration, speed and wheel turning have been determined in Chapter 4. Standard value for acceleration is 1 m/s^2. Standard high speed on the berth lanes and the yard lanes is 3 m/s and standard low speed on the warehouse lanes is 1 m/s. The time needed for a rotation disk to turn the wheels of an rgv is 4 seconds.

The number of Rgv's per portainer
The standard number rgv's subscribed to a portainer during unloading and fast loading differs from the number of rgv's subscribed to a portainer during adjacent loading and shuffling. In the first case 6 rgv's per portainer is the standard amount, while in the latter case 10 rgv's per portainer is standard. These numbers are based on the rgv cycle times as determined in Appendix 10 and the necessity to keep the portainers from waiting for rgv's.

Number of Warehouse cranes
The standard number of warehouse cranes equals the maximum number of warehouse cranes possible on the terminal. This amounts to 70 warehouse cranes of which 30 warehouse cranes are located in the stacks behind Berth 2. Berth 1 and Berth 3 each host 20 warehouse cranes. After the simulation runs the occupancy rates of the warehouse cranes will be analysed.

Handling performance of the Warehouse cranes
As opposed to the portainers, the warehouse cranes have an invariable handling performance in the simulation program. The warehouse crane performance is determined by the average time that a crane needs to work in order to retrieve a container from the stack and delivering it back to a service point. This average crane cycle is calculated in Chapter 4.3 and is 80 seconds. One simulation run will be made by using variable crane cycles.

8.2 Performance indicators

The simulation runs with one and three vessels will be evaluated using a number of performance indicators. A wide variety of simulation runs will be carried out during each of the two types of berth occupation. The simulation of one feeder vessel moored at Berth 1, for instance, consists of four separate runs. The runs simulate unloading, fast loading, adjacent loading and shuffling. By performing a separate run for each mode the output of each simulation will be easier to analyse.

The performance indicators that will be used for this purpose are listed below:

- Average container production per berth
- Portainer occupation rates
- Warehouse crane occupation rates
- Rgv rows on crossing nodes
- Rgv rows on the warehouse lanes
- Rgv rows under the portainers

Average container production per berth
By determining the container production during each run a first impression can be obtained on the performance of the terminal during each mode. The container production during a run with a high berth occupation can be compared to runs with a single vessel.

Portainer occupation rates
Apart from shuffling, all the vessel handling modes make use of portainers. The portainer occupancy therefore is an important performance indicator. The portainer occupancy rate must be kept as high as possible. When the average portainer cycletimes stay the same, the portainer occupancy rate is directly linked to the portainer production.

Warehouse crane occupancy rates
The warehouse crane occupancy rate is especially important for the determination of the number of cranes in the stack.

Rgv rows on crossing nodes
Rgv rows should be minimised.

Rgv rows on the warehouse lanes
The length of the rgv row on a warehouse lane in a warehouse may not exceed the number of 3 rgv's because there is no more room available in a stack-row.

Rgv rows under the portainers
Rgv rows under portainer may preferably not exceed 6 rgv's, as there is not enough room for more. This number can only be exceeded if additional lanes are added to the layout.

8.3 Feeder vessel moored at Berth 1

With just one vessel present at the terminal and no interference from rgv activities operating for other berths the handling productivities will turn out to be relative high. First a feeder vessel moored at Berth 1 will be analysed during the different handling procedures, followed by the handling procedures of a mega vessel moored at Berth 2.

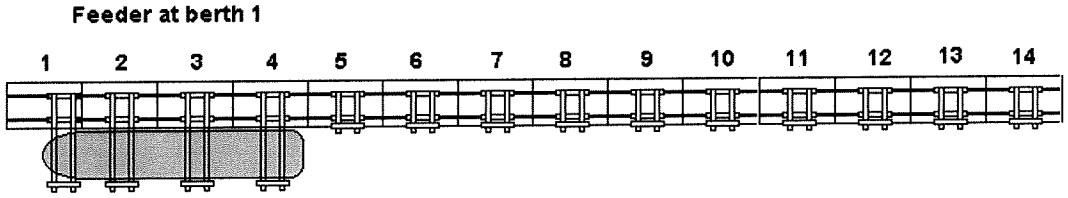


Figure 8.1, a feeder vessel moored at Berth 1

The next simulation runs have been carried out with one feeder vessel moored at Berth 1. Berth 2 and Berth 3 remain empty. The undisturbed simulation runs for a feeder vessel, moored at Berth 1 are assumed to be similar to those of a feeder vessel moored at Berth 3.

The four experiments with a feeder vessel moored at Berth 1 are:

- Unloading of a feeder
- Fast-loading of a feeder
- Loading of a feeder from an adjacent berth
- Shuffling from a feeder berth

The basic parameters used for each of the experiments are shown in Appendix 13-15.

8.3.1 Unloading of a feeder vessel

This simulation run concerns the unloading of 1050 containers from the feeder vessel to the stack-clusters located behind Berth 1. In Appendix 13 the rgv waiting rows under the portainers, at the nodes and on the warehouse lanes can be seen.

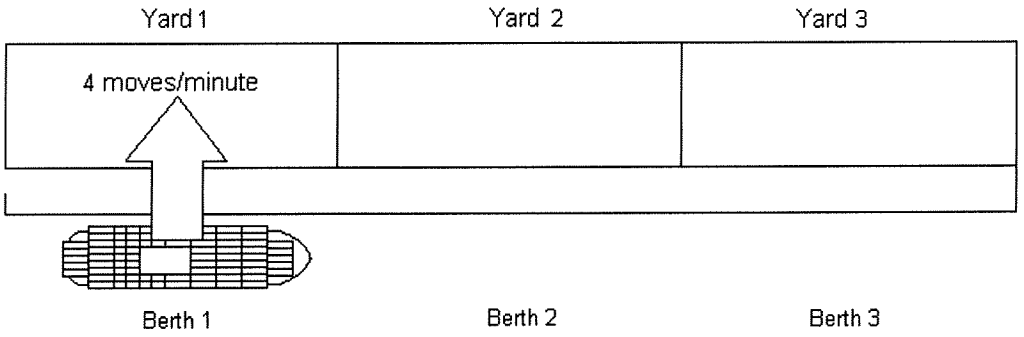


Figure 8.2, the unloading of a feeder vessel

Portainer occupancy rates
The portainer occupancy rates during the unloading of the feeder vessel are shown in Figure 8.3. These occupancy rates will act as a comparison with other experiments. Each portainer has 6 rgv's subscribed during the unloading of the feeder vessel.

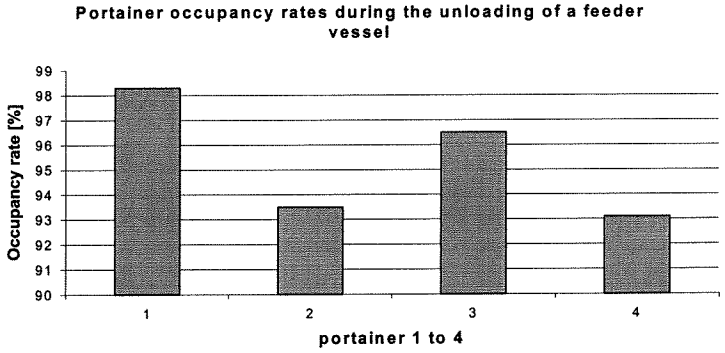


Figure 8.3, The portainer occupancy rates during the unloading of a feeder vessel at Berth 1

The occupancy rate of the odd numbered portainers is higher than the occupancy rate of the even numbered portainers. This is a result of the relative longer distance that the rgv's have to cover at the even portainer locations resulting in a longer rgv driving cycle.

Handling production
The handling production of the portainers located at Berth 1 during the unloading of one feeder vessel is 237 moves per hour. This results in 355 TEU per hour. The handling time for a feeder vessel during unloading is 4.4 hours.

Rgv waiting queues under the portainers
Figure 8.4 shows the waiting queues of the rgv's under each portainer. The portainer occupancy rates plotted in Figure 8.3 are very high, which agrees with the fact that the rgv queue under the portainers is seldom empty.

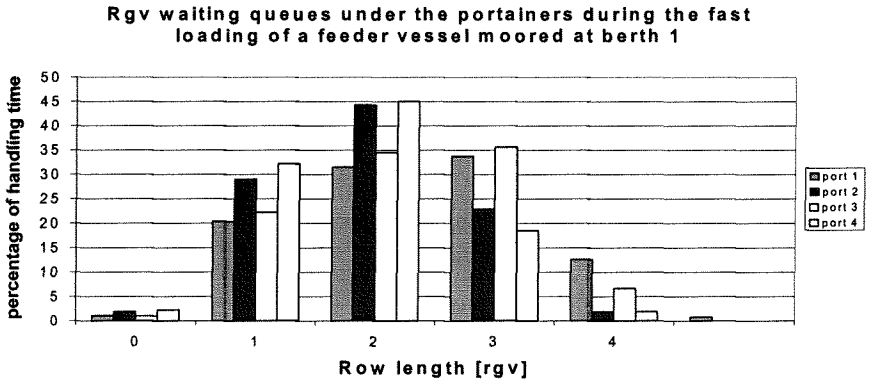


Figure 8.4, rgv queue under the portainers at Berth 1 during the unloading of a feeder

The rgv waiting queues under the portainers almost never exceed the length of 4 rgv's. The average queue length is 1.5 rgv's.

Rgv waiting queues on the warehouse lanes
The figures of the queues on the warehouse lanes show no extraordinary results. The queues almost never exceed 1 rgv in length. The queues on the warehouse lanes are given in Appendix 13. An example of a rgv waiting queue on a warehouse lane during the unloading and subsequently the fast loading of the feeder vessel can be seen in Figure 8.5.

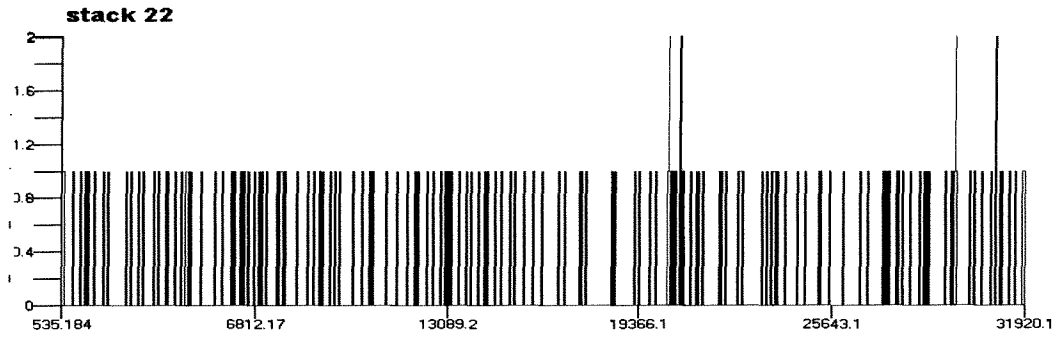


Figure 8.5, rgv waiting queues on warehouse lane 22

Warehouse crane occupancy rates
The warehouse crane occupancy rates for the warehouse cranes located in Yard 1 have been calculated in Appendix 13, resulting in:

Mean occupancy rate for a warehouse crane located in the 20' stack is 33.1%.
Mean occupancy rate for a warehouse crane located in the 40' stack is 22.1 %.

These occupancy rates are acceptable. It might be possible to operate with fewer warehouse cranes.

8.3.2 Fast loading of a feeder vessel

This simulation run concerns the fast loading of 1050 containers onto the feeder vessel from the stack-clusters located behind Berth 1. The fast loading of a feeder vessel is only possible if the containers have been shuffled from Yard 2 to Yard 1 earlier. In Appendix 13 the rgv waiting rows concerning the portainers, nodes and warehouse lanes can be seen.

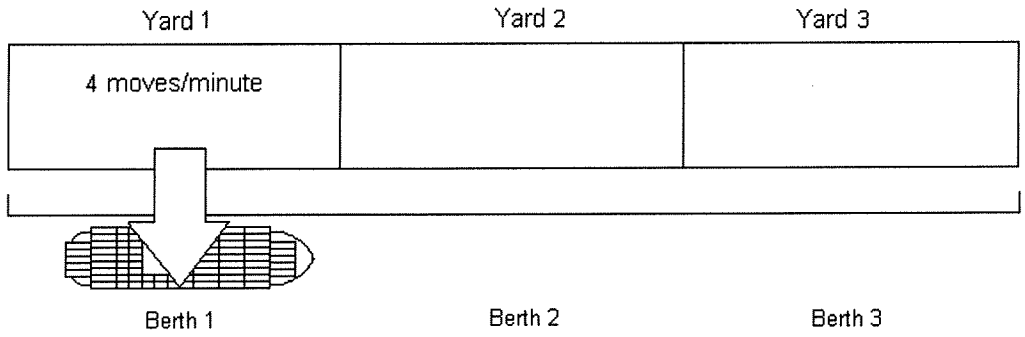


Figure 8.6, the fast loading of a feeder vessel

Portainer occupancy rates
If a feeder vessel is being loaded with containers from the yard behind the berth the feeder vessel is moored to, the rgv's use the same routes as during the unloading of a feeder vessel. The expected portainer productions will therefore be more or less the same. The portainer occupancy rates are given in Figure 8.7.

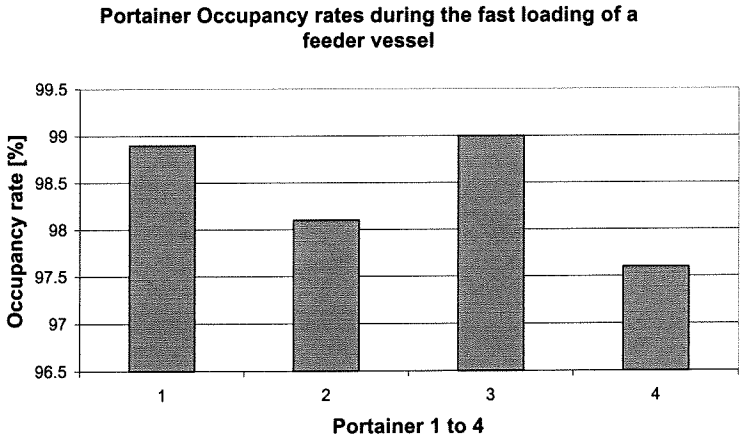


Figure 8.7, the portainer occupancy rates during the fast loading of a feeder vessel

Handling production
The handling production of the portainers located at Berth 1 during the fast loading of one feeder vessel is 244 moves per hour. This results in 366 TEU per hour. The handling time for a feeder vessel during unloading is 4.3 hours. The portainer productions are a bit higher than when unloading a feeder vessel.

Rgv waiting queues under the portainers
Figure 8.8 shows the waiting queues of the rgv's under each portainer. The portainer occupancy rates plotted in Figure 8.7 are high, which agrees with the fact that the rgv queue under the portainers is seldom empty.

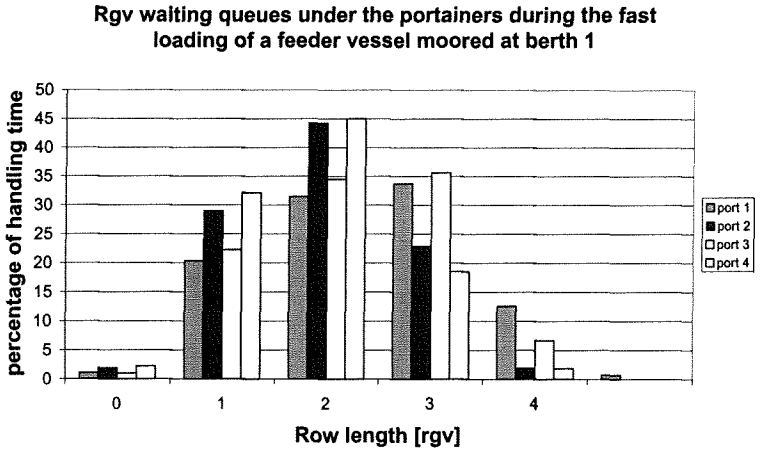


Figure 8.8, rgv waiting rows under portainers at Berth 1 when loading fast

Warehouse crane occupancy rates
The warehouse crane occupancy rates for the warehouse cranes located in Yard 1 have been calculated in Appendix 13, resulting in:

Mean occupancy rate for a warehouse crane located in the 20' stack is 33.9 %.
Mean occupancy rate for a warehouse crane located in the 40' stack is 22.6 %.

These occupancy rates seem acceptable. It might even be possible to operate with fewer warehouse cranes.

8.3.3 Adjacent loading of a feeder

The adjacent loading of a feeder vessel moored at Berth 1 means that the 1050 to be loaded containers have to be retrieved from Yard 2. 10 rgv's per portainer will be used for this. In this run the node rows on the terminal have been analysed, as the rgv's are to travel between the berths. In Appendix 14 the rgv waiting rows concerning the portainers, nodes and warehouse lanes can be seen for this run.

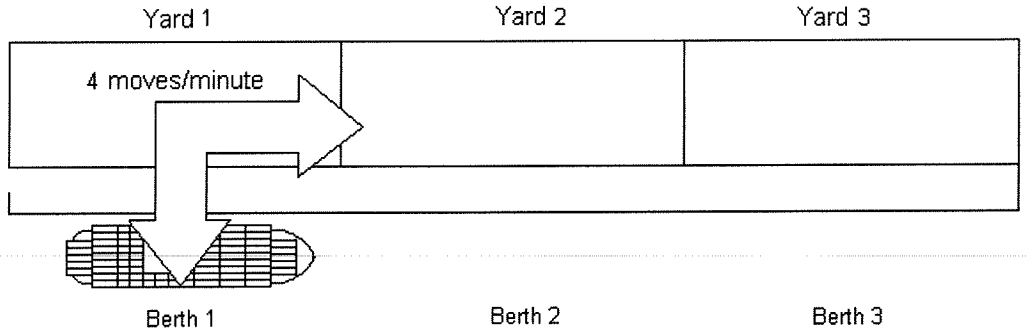


Figure 8.9, the adjacent loading of a Feeder

Portainer occupancy rates
The portainer occupancy rate during the adjacent loading of a feeder vessel is higher as the portainers are situated closer to Yard 2. This is a result of the shorter driving distances of the rgv's.

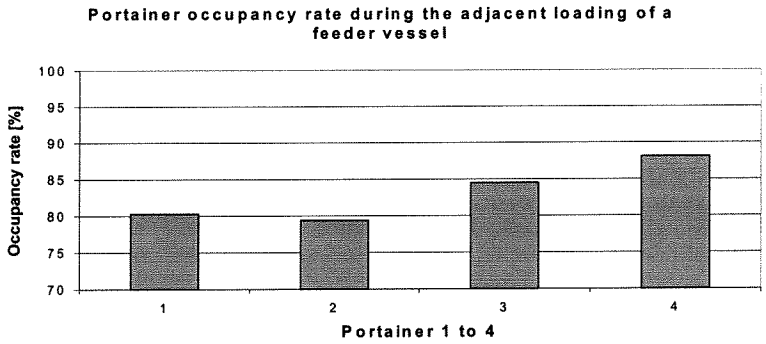


Figure 8.10, the portainer occupancy rates when loading a feeder from an adjacent berth

Higher portainer occupancy rates might be achieved by increasing the number of rgv's per portainer. Especially Portainer 1 & 2 will have their production increased by this measure.

Handling production
The portainer production is 51.5 moves per portainer per hour during adjacent loading. This results in a berth production of 206 moves per hour. This results in 309 TEU per hour per feeder berth. The handling time needed for the adjacent loading of 1050 containers is 5.1 hours.

Rgv waiting queues under the portainers
The length of the waiting queues under the portainers rarely exceeds the length of 4 rgv's, as can be seen in Figure 8.11.

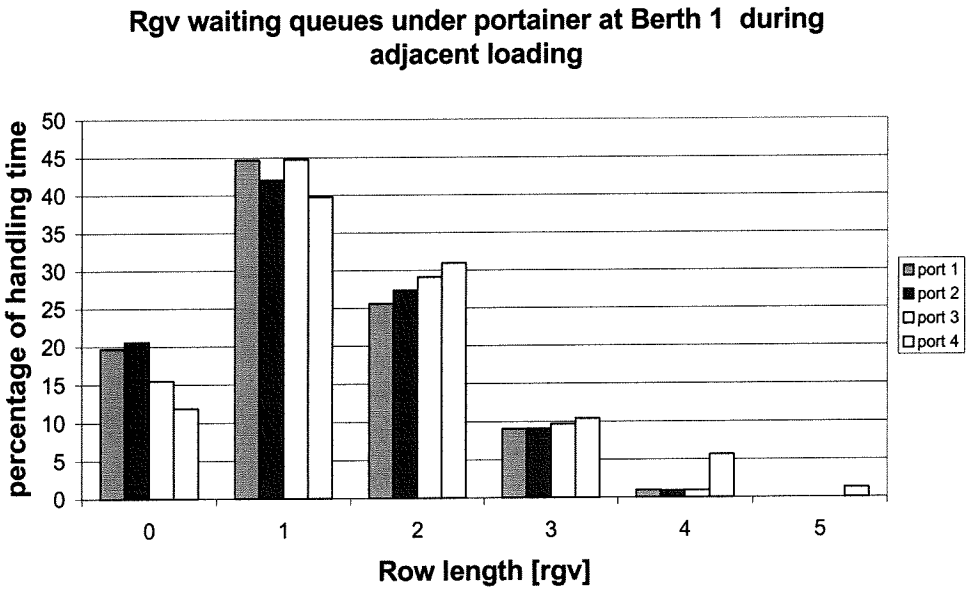


Figure 8.11, the rgv waiting queues under the portainers at Berth 1 when adjacent loading

At the portainer locations 1 and 2, the waiting rows will be empty for about 20 percent of the time. In this case it could be an option to work with more rgv's per portainer.

Rgv waiting queues at the nodes
The nodes containing the longest rgv queues are the nodes 116, 114, 260, 290, 243 and 273. The queues at these nodes are shown in Appendix 14. The length of the queues at these nodes is generally 1 or 2 rgv's. A node queue rarely contains 3 rgv's.

Rgv waiting queues on the warehouse lanes
The queues on the warehouse lanes in yard 2 are at the most 2 rgv's in length but most times just 1. These queues are shown in appendix 14.

Warehouse crane occupancy rates
The warehouse crane occupancy rates for the warehouse cranes located in Yard 2 have been calculated in Appendix 14 resulting in:

Mean occupancy rate for a warehouse crane located in the 20' stack is 19.1 %.
Mean occupancy rate for a warehouse crane located in the 40' stack is 12.7 %.

These occupancy rates seem acceptable. It might even be possible to operate with fewer warehouse cranes.

8.3.4 Shuffling from a feeder berth

When the feeder vessel has been unloaded and loaded again the rgv's operating for the portainers on Berth 1 will be subscribed to the stack-clusters behind their portainers. The containers delivered by the feeder vessel can then be shuffled from Yard 1 to Yard 2. The number of shuffled containers is 1250.

When the rgv unloads a container from Yard 1 into Yard 2 the rgv will wait to be loaded with a container that can be shuffled from Yard 2 to Yard 1, if there is any. Otherwise it will return empty to retrieve the next shuffle container from Yard 1. In Appendix 15 the rgv waiting rows concerning the nodes and warehouse lanes can be seen for this run.

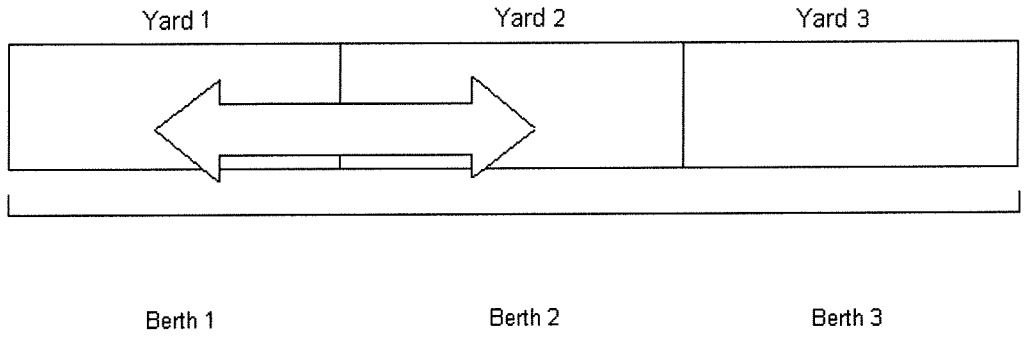


Figure 8.12, the shuffling from Yard 1

Shuffle production
During this simulation run 1250 containers have been placed equally divided over the 20' stack and the 40' stack in Yard 1. The same amount of containers has been placed in Yard 2.

The shuffle production is defined as the total amount of shuffled containers per time unit. The total amount of shuffled containers is:

Number of shuffled containers from Yard 1 to Yard 2:	1,250
Number of shuffled containers from Yard 2 to Yard 1:	1,250
Total amount of shuffled containers on the terminal:	2,500

The time necessary to shuffle this amount of containers is 26,670 seconds or 7.4 hours.

This results in a total shuffle production of: $2{,}490 / 7.4 = 337.8$ containers per hour, or 5.6 containers per minute.

Rgv waiting queues at the nodes
The nodes with the highest expected rgv rows have been analysed in order to get an impression on the busyness on the terminal during the shuffling procedure. The nodes are the same ones as during the adjacent loading of a feeder vessel. The nodes are nodes 116, 114, 260, 290, 243 and 273. The waiting rows of these nodes have been plotted in Appendix 15. What can be seen is that length of the rgv queues occasionally exceeds 2 rgv's. The node with the longest queue has been plotted in Figure 8.13.

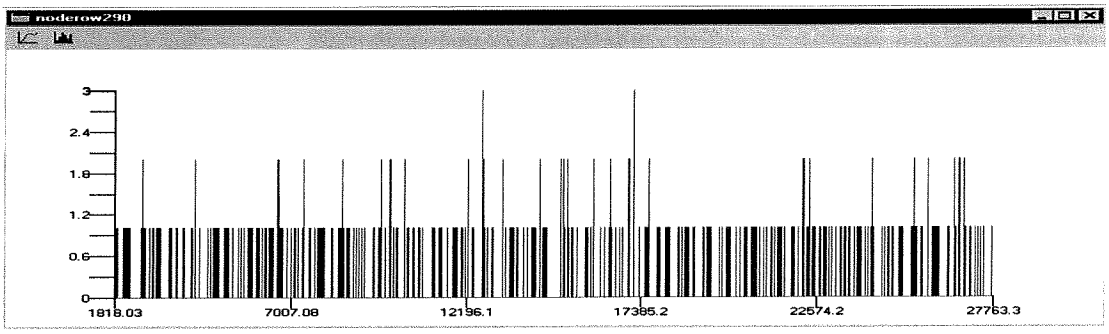


Figure 8.13, rgv queue at node 290

Waiting rows on the warehouse lanes
The warehouse lanes in the warehouses located in Yard 1 and Yard 2 have been analysed. The warehouse lanes in Yard 1 however will be visited more frequently as the number of warehouses is smaller compared to Yard 2.

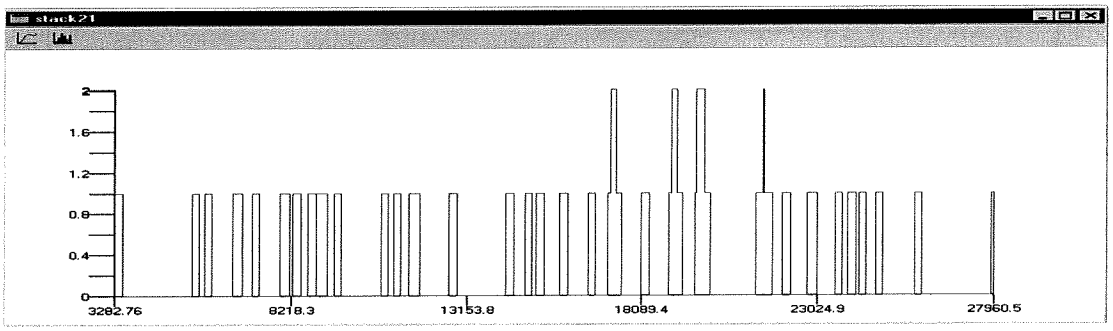


Figure 8.14, rgv waiting row on warehouse lane 21

Warehouse lane 21 is located in Yard 1. Despite the frequent visits of the rgv's the row length does not reach 3 rgv's.

Warehouse crane occupancy rates
The warehouse crane occupancy rates for the warehouse cranes located in Yard 1 are the highest on the terminal during the shuffling from Berth 1. The occupancy rates of these warehouse cranes have been calculated in Appendix 15, resulting in:

Mean occupancy rate for a warehouse crane located in the 20' stack in Yard 2 is 46.7 %
Mean occupancy rate for a warehouse crane located in the 40' stack in Yard 2 is 31.2 %

These occupancy rates are acceptable.

8.3.5 Recapitulation of the feeder vessel simulation runs

Handling rates
The performance of the terminal during the handling of a feeder vessel moored at Berth 1 without any disturbances from activities on other berths has been plotted in Table 8.1.

	Number of containers	Production per portainer [moves/hour]	Production per berth [moves/hour]	Production per berth [TEU/hour]	Service times [hours]
Unloading	1050	59	236	355	4.5
Fast loading	1050	61	244	366	4.3
Adjacent loading	1050	51	206	309	5.1

Table 8.1, the undisturbed container productions for Berth 1

The assumption that the unloading of a feeder vessel can be carried out with comparable speed as during the fast loading procedure has proven to be correct. The adjacent loading of a feeder vessel takes about 13 % longer than fast-loading with this terminal configuration. The most time consuming unload-load combination for the feeder vessel is the unloading combined with the adjacent loading strategy. The undisturbed service time for the handling of a feeder vessel is: 4.5 + 5.1 = 9.6 hours.

Shuffling containers from Yard 2 to Yard 1 allows the feeder vessel to be fast-loaded. The service time will then be 4.5 + 4.3 = 8.8 hours. This is a reduction of 0.8 hours or 8 % of the total service time compared to the adjacent loading variant. The production during shuffling has been determined to amount to 337.8 containers per hour. This is based on shuffling 2 containers per rgv cycle. The first container is shuffled from Yard 1 to Yard 2 while the second will be shuffled from Yard 2 to Yard 1. This makes the determined shuffle production an optimistic one. Even so, it is questionable if a reduction of 0.8 hours in the service times will justify the shuffling procedure.

Operational rgv's - Portainer occupancies
The amount of operational rgv's on the reference terminal during the different procedures has been established in Chapter 5 to be 6 rgv's per portainer for the unloading and fast-loading procedure and 10 rgv's per portainer for the adjacent loading procedure. Having carried out the simulation runs it can be concluded that the chosen 6 rgv's for the unloading and fast-loading procedures results in high portainer occupancy rates of around 95 % without creating excessive waiting rgv rows in the stacks, nodes or under the portainers. Row length under the portainers can be seen in Appendix 13-15. These row lengths seldom exceed the length of 5 rgv's during any of the simulated handling procedures.

The adjacent loading procedure with 10 rgv's per portainer, results in a mean portainer occupancy rate of 83 % on Berth 1. Increasing the number of rgv's per portainer to 12 should increase the occupancy rates and reduce the service time of a feeder vessel, thereby further justifying the criticism on the shuffling procedure.

Warehouse crane occupancy rates
The occupation rates of the warehouse cranes during the undisturbed handling of a feeder vessel have been shown in Table 8.2.

		Procedure			
		Unloading to Yard 1 (%)	Fast-loading from Yard 1 (%)	Adjacent loading from Yard 2 (%)	Shuffling from Yard 1 (%)
Stack type	20' stack	33.1	33.9	19.1 (Y2)	46.7 (Y1)
	40' stack	22.1	22.6	12.7 (Y2)	31.3 (Y1)

Table 8.2, warehouse crane occupation rates during the undisturbed handling of a feeder vessel

The warehouse crane occupancy rates are low. During the undisturbed handling of a feeder vessel it is recommend operating with less warehouse cranes than have been introduced on the reference terminal. In Chapter 8.5 more definite conclusions can be made as that chapter concerns the handling of 3 vessels moored at the terminal.

Rgv waiting queues on the nodes
Rgv row length on the nodes during adjacent loading has been plotted in Appendix 14. What can be concluded is that during that simulation run the row lengths on the nodes never exceeds the length of 2 rgv's.

8.4 Experiment: Mega vessel moored at Berth 2

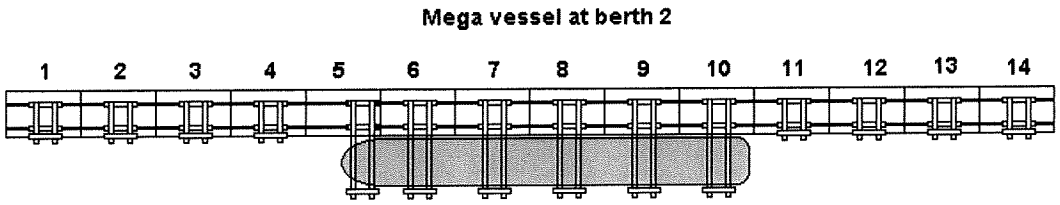


Figure 8.15, mega vessel moored at Berth 2.

When there is no disturbance from other activities on the terminal the handling productions are expected to be high. The next simulation runs have been carried out with one mega vessel moored at Berth 2. Berth 1 and Berth 3 remain empty.

The four experiments with the mega vessel are:

- Unloading of a mega vessel
- Fast-loading of a mega vessel
- Loading of the mega vessel from the adjacent berths
- Shuffling from Berth 2

The basic parameters used during these runs are as shown in Appendix 16-19.

8.4.1 The unloading of the mega vessel

This simulation run concerns the unloading of a mega vessel moored at Berth 2. 3150 containers will be unloaded and stacked in Yard 2, located behind Berth 2. In Appendix 16 the rgv waiting rows concerning the portainers, nodes and warehouse lanes can be seen.

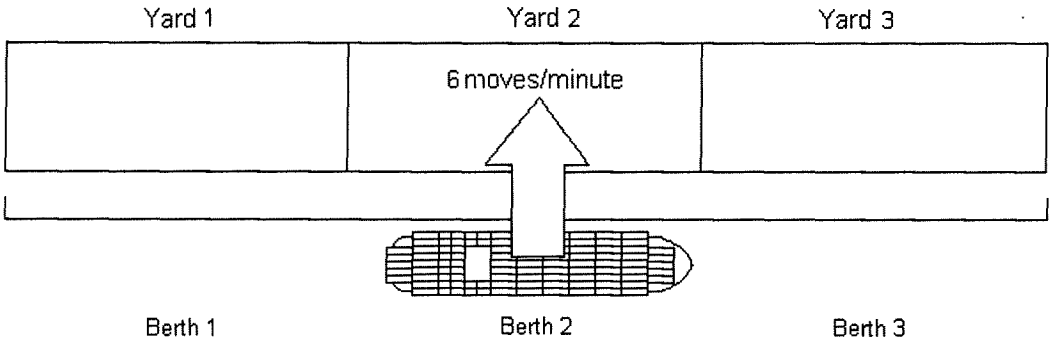


Figure 8.16, the unloading of a mega vessel

Portainer occupancy rates
The portainer occupancy rates during the unloading of a mega vessel using 6 rgv's per portainer are shown in Figure 8.17. These occupancy rates will act as a comparison for other experiments using a mega vessel in the future.

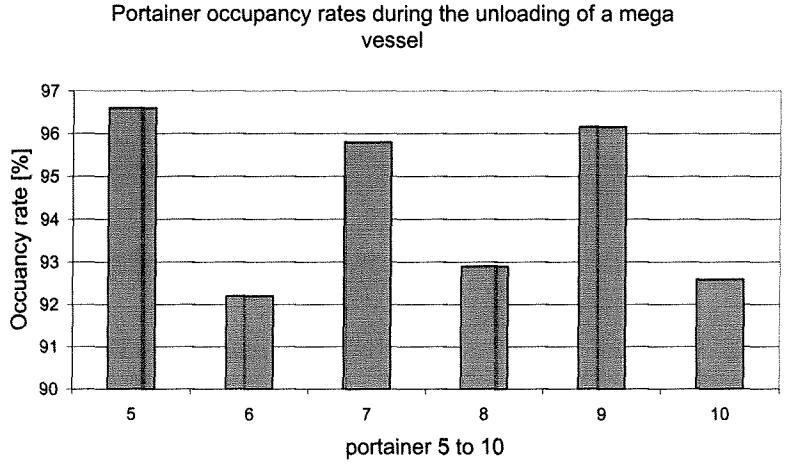


Figure 8.17, the occupancy rate of portainers 5 to 10

The occupancy rate of the odd numbered portainers is higher than the occupancy rates of the even numbered portainers. This is a result of the relative longer distance the rgv's have to cover when subscribed to the even numbered portainers resulting in a longer rgv driving cycle.

Handling production
The handling production of the portainers located at Berth 2 during the unloading is 58.7 moves per portainer per hour. This results in 352 moves per berth per hour or 528 TEU per berth per hour. The handling time for the unloading of a mega vessel is 8.9 hours.

Rgv waiting queues under the portainers
Figure 8.18 shows the waiting queues of the rgv's under each portainer. The portainer occupancy rates plotted in are very high, which agrees with the fact that the rgv queue under the portainer is seldom empty.

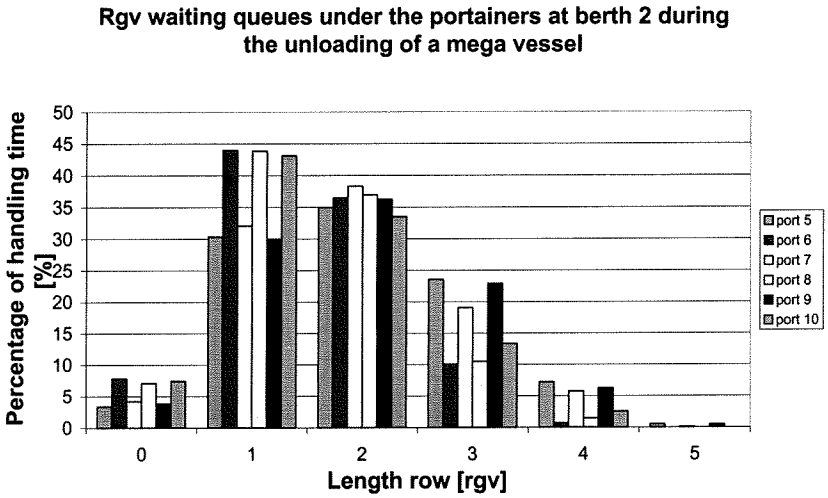


Figure 8.18, rgv waiting queues under the portainers on Berth 2

The rgv waiting queues under the portainers almost never exceed the length of 4 rgv's. The average queue length is 1.55 rgv's.

Rgv waiting rows on the warehouse lanes
The figures of the queues on the warehouse lanes show no extraordinary results. The queues almost never exceed 1 rgv in length. The queues on the warehouse lanes are given in Appendix 16.

Warehouse crane occupancy rates
The warehouse crane occupancy rates for the warehouse cranes located in Yard 2 have been calculated in Appendix 16, resulting in:

Mean occupancy rate for a warehouse crane located in a 20' stack is 32.8%
Mean occupancy rate for a warehouse crane located in a 40' stack is 21.8%

These occupancy rates seem acceptable. It might even be possible to operate with fewer warehouse cranes. The queues on some of the warehouse lanes are given in Appendix 16.

8.4.2 The fast loading of a mega vessel

This simulation run concerns the fast loading of 3150 containers onto a mega vessel from Yard 2. The routes of the rgv's are almost the same as during unloading. The results are expected to be comparable. In Appendix 17 the rgv waiting rows concerning the portainers, nodes and warehouse lanes can be seen.

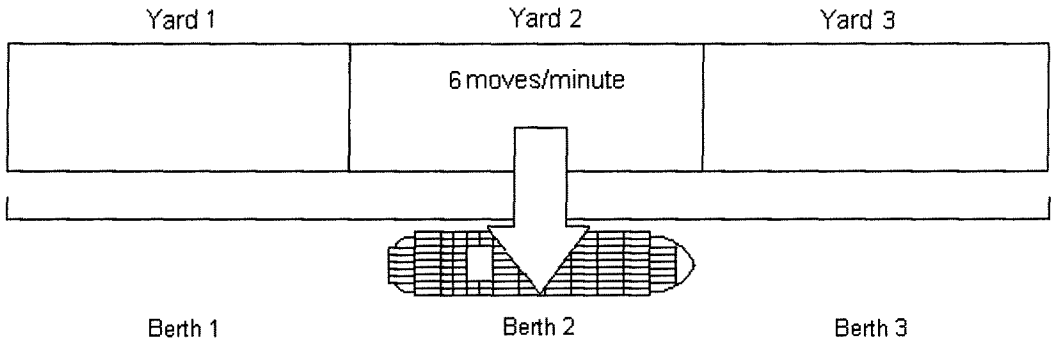


Figure 8.19, the fast loading of a mega vessel

Portainer occupancy rates
In Figure 8.20 the portainer occupancy rates when fast loading a mega vessel are given. The difference in driving distance of the rgv's subscribed to the even numbered and odd numbered portainers is apparent.

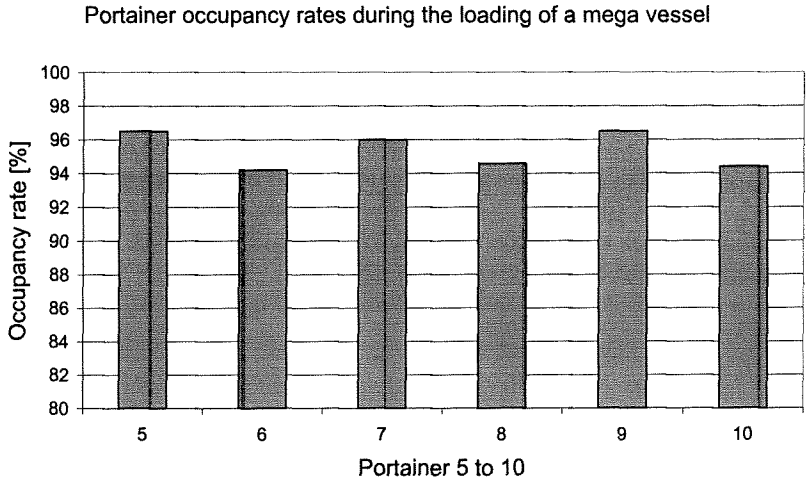


Figure 8.20, Portainer occupancy rates during the fast loading of a mega vessel.

Handling production
The production of Berth 2 during the fast loading of 3150 containers onto a mega vessel is 350 moves per hour. This results in 525 TEU per berth per hour. The required handling time is 9 hours.

Rgv waiting queues under the portainers
Figure 8.21 shows the waiting queues under the portainers during the fast loading the mega vessel. The waiting queues under the portainer are seldom empty.

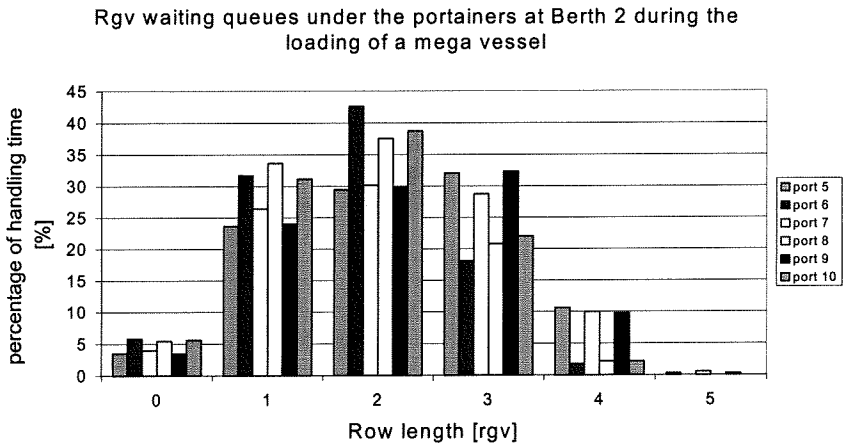


Figure 8.21, the rgv waiting queues under the portainers when loading a mega vessel

Warehouse crane occupancy rates
The warehouse crane occupancy rates for the warehouse cranes located in Yard 2 have been calculated in Appendix 17, resulting in:

Mean occupancy rate for a warehouse crane located in a 20' stack is 32.8%
Mean occupancy rate for a warehouse crane located in a 40' stack is 21.8%

These occupancy rates seem acceptable. The queues on some of the warehouse lanes are given in Appendix 17.

8.4.3 Adjacent loading of a mega vessel

The adjacent loading of a mega vessel moored at Berth 2 means 3150 containers will have to be retrieved from Yard 1 & 3. The rgv waiting rows on the nodes on the terminal have been analysed, as the rgv's travel between the yards. In Appendix 18 the rgv waiting rows concerning the portainers, nodes and warehouse lanes can be seen for this run.

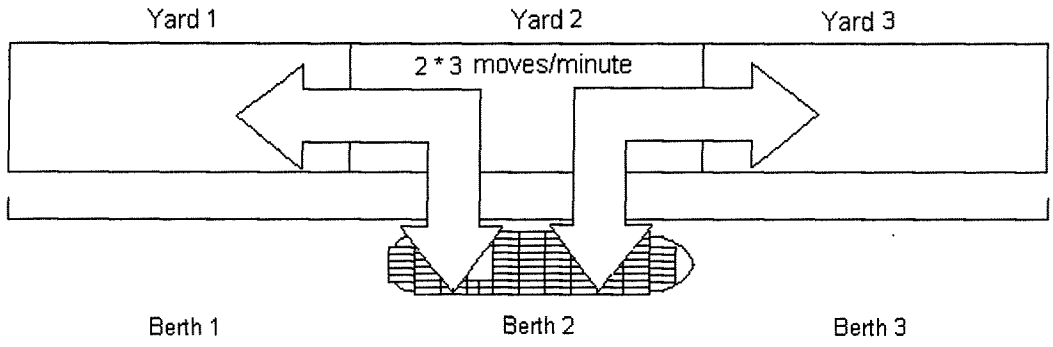


Figure 8.22, the adjacent loading of a mega vessel

Portainer occupancy rates
During this simulation run 10 rgv's per portainer are used.

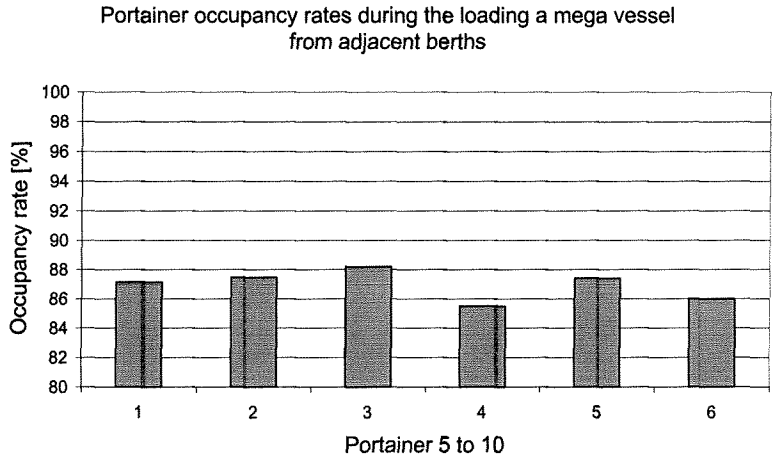


Figure 8.23, Portainer occupancy rate when loading a mega vessel from adjacent berths

Handling production
The production per portainer is 51 moves. Per berth this results in 305 moves per hour. This results in 458 TEU per hour per berth. The handling time needed for the undisturbed adjacent loading is 10 hours.

Rgv waiting queues under the portainers
Figure 8.24 shows that the rgv waiting queues under the portainers sometimes exceed the length of 4 rgv's.

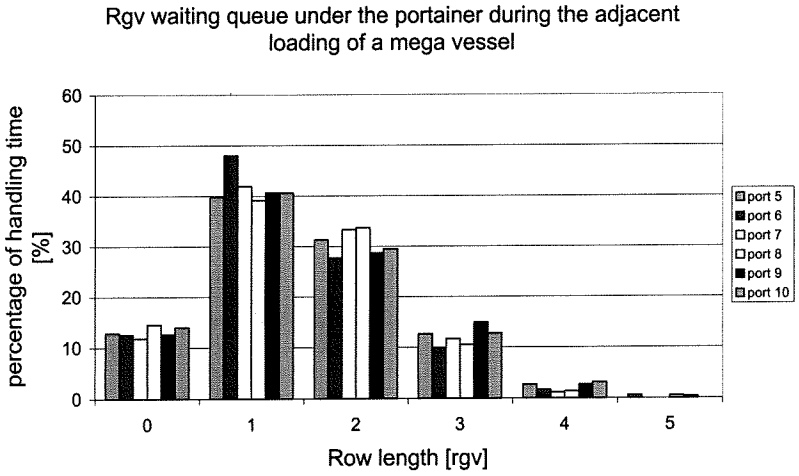


Figure 8.24, the rgv waiting queues under the portainers during the loading from an adjacent berth

At all the portainer locations the waiting row is empty about 13% of the time. In this case it could be an option to work with more rgv's per portainer.

Rgv waiting queues at the nodes
The nodes containing the longest rgv queues are the nodes that connect the inter-berth lanes with the stacks. The queues at these nodes are shown in Appendix 18.

Warehouse crane occupancy rates
The warehouse crane occupancy rates for the warehouse cranes located in Yard 2 have been calculated in Appendix 18, resulting in:

Mean occupancy rate for a warehouse crane located in a 20' stack is 22.6%
Mean occupancy rate for a warehouse crane located in a 40' stack is 15.0%

These occupancy rates seem acceptable. The queues on some of the warehouse lanes are given in Appendix 18.

8.4.4 The shuffling from Berth 2

When the mega vessel has been unloaded and loaded again the rgv's operating for the portainers on Berth 2 will be subscribed to the stack-clusters behind their portainers. The 3150 containers delivered by the mega vessel will then be shuffled from Yard 2 to Yard 1 & 3. When the rgv transports a container from Yard 2 into Yard 1 or Yard 3, the rgv will wait to be loaded by a container that can be shuffled from Yard 1 or 3 to Yard 2, if there is any. Otherwise it will return empty to retrieve the next shuffle container from Yard 2. In Appendix 19 the rgv waiting rows concerning the nodes and warehouse lanes can be seen for this run.

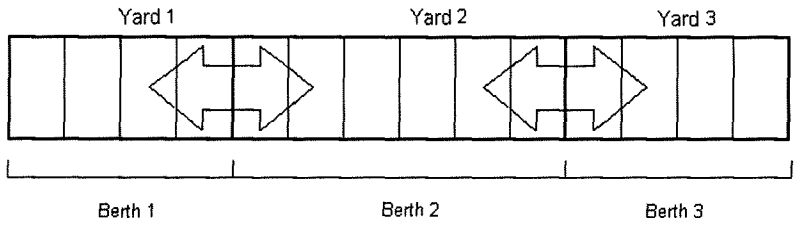


Figure 8.25, shuffling from Berth 2

Shuffle productions
The shuffle production is defined as the total amount of shuffled containers per time unit. The shuffle procedure lasted 720 minutes, 11.25 hours. In this time interval rgv's moved 1575 containers from Yard 2 to Yard 1, 1575 containers from Yard 2 to Yard 3 and 3150 containers from Yard 1 and 3 to Yard 2. In total 6300 containers were put in their place.

This results in a total shuffle production of: $6300/11.25 = 560$ containers per hour, or 9.3 containers per minute.

Rgv waiting queues at the nodes
If a mega vessel has left Berth 2, the shuffling procedure can start. While shuffling from Berth 2 to the adjacent berths 10 rgv's per stack-cluster are used. In this experiment 10 * 6 = 60 rgv's are used. It is expected that this should be no problem for the terminal and no excessive queues should develop at any node or stack. The queues at the nodes almost never exceed the length of 3 rgv's except at node 363 and 393. These nodes are shown in Figure 8.26 and Figure 8.27. These nodes are the connections between the bottom inter berth lanes and the yard lanes of Yard 2. As can be seen in Appendix 2.

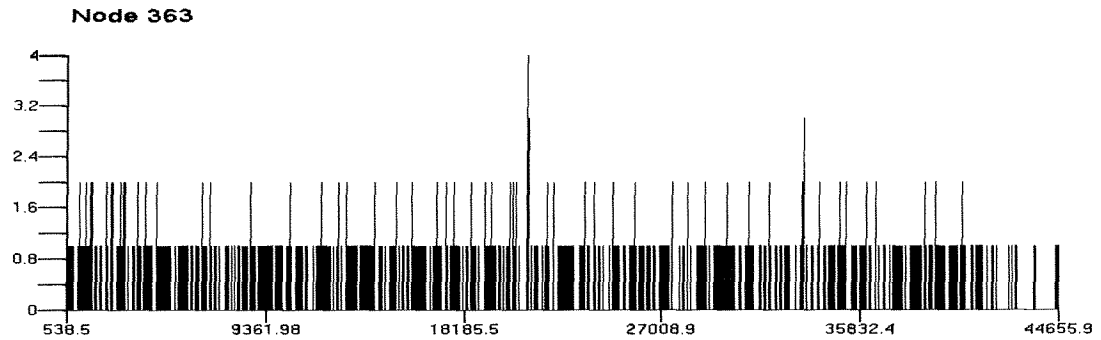


Figure 8.26, the rgv queue at node 363 when shuffling from Berth 2

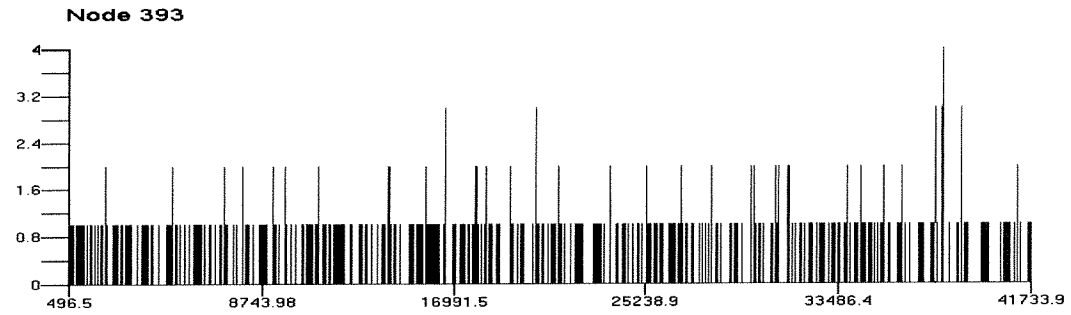


Figure 8.27, the rgv queue at node 393 during shuffling from Berth 2

Some other nodes are shown in Appendix 19.

The queues on the warehouse lanes
In this experiment in total 60 shuffle rgv's get their containers from 80 warehouse lanes, so no problems are expected on the warehouse lanes. The queues on the warehouse lanes almost never exceed the total of 2 rgv's. Warehouse lane 23 is an exception the queues on this warehouse lane can be seen in Figure 8.28. The queues of stack 21 to 30 are shown in Appendix 19.

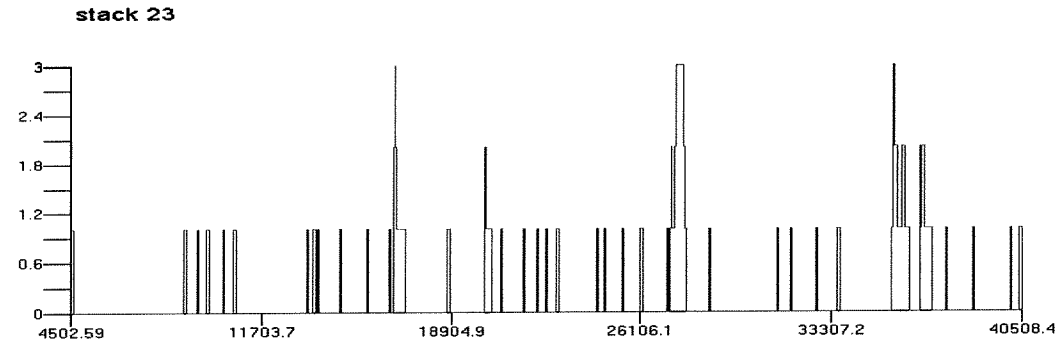


Figure 8.28, the rgv queues on warehouse lane 23 when shuffling from Berth 2

Warehouse crane occupancy rates
The warehouse crane occupancy rates for the warehouse cranes located in Yard 2 are the highest on the terminal during the shuffling from Berth 2. The occupancy rates of these warehouse cranes have been calculated in Appendix 19, resulting in:

Mean occupancy rate for a warehouse crane located in the 20' stack in Yard 2 is 51.9 %
Mean occupancy rate for a warehouse crane located in the 20' stack in Yard 2 is 34.6 %

These occupancy rates are acceptable.

8.4.5 Recapitulation of the mega vessel simulation runs

Handling rates
The performance of the terminal during the handling of a mega vessel moored at Berth 2 without any disturbances from activities on other berths has been plotted in Table 8.3.

	Number of containers	Production per berth [moves/hour]	Production per portainer [moves/hour]	Production per berth [TEU/hour]	Service times [hours]
Unloading	3150	352	58	528	8.9
Loading	3150	350	58	525	9
Adjacent loading	3150	324	54	486	9.7

Table 8.3, berth productions for Berth 2

As was the case with the feeder vessel, the assumption that the unloading of a feeder vessel can be carried out with comparable speed as the fast loading has proven to be correct. The adjacent loading of a mega vessel takes about 8 % longer than during fast loading with this terminal configuration. The most time consuming unload-load combination for the mega vessel is the unloading combined with the adjacent loading strategy. The undisturbed service time for the handling of a mega vessel is 8.9 + 9.7 = 18.6 hours.

Shuffling containers from Yard 1 & 3 to Yard 2 will allow the mega vessel to be fast-loaded. The service time will then be 8.9 + 9 = 17.9 hours. This is a reduction of 0.7 hours or 4 % compared to the handling variant with adjacent loading. The production during shuffling has been determined to amount to 560 containers per hour. This is based on shuffling 2 containers per rgv cycle. The first container is shuffled from Yard 2 to Yard 1 or 3 while the second will be shuffled back to Yard 2 from Yard 1 or Yard 3. This makes the determined shuffle production an optimistic one. Even so, it is questionable if a reduction of 0.7 hours in the service times will justify the shuffling procedure.

Operational rgv's - Portainer occupancies
The amount of operational rgv's on the reference terminal during the different handling procedures for a mega vessel has been established to be 6 rgv's per portainer for the unloading and fast-loading procedure and 10 rgv's per portainen for the adjacent loading procedure in Chapter 5. Having carried out the simulation runs it can be concluded that the chosen 6 rgv's for the unloading and fast-loading procedures results in high portainer occupancy rates without creating excessive rgv waiting rows in the stacks or on the nodes. The adjacent loading procedure with 10 rgv's per portainer however results in a mean portainer occupancy rate of about 83 % on Berth 2. Increasing the number of rgv's per portainer to 12 should increase the occupancy rates and reduce the service time of a mega vessel. Obviously bigger queues under the portainers can be expected.

Warehouse crane occupancy rates
The occupation rates of the warehouse cranes during the undisturbed handling of a mega vessel have been shown in Table 8.4.

		Procedure			
		Unloading to Yard 2 (%)	Fast-loading from Yard 2 (%)	Adjacent loading from Yard 1 or 3 (%)	Shuffling from Yard 2 (%)
Stack type	20' stack	32.8	32.8	22.6 (Y1 or Y3)	51.9 (Y2)
	40' stack	21.8	21.8	15.0 (Y1 or Y3)	34.6 (Y2)

Table 8.4, warehouse crane occupation rates during the undisturbed handling of a mega vessel

The warehouse crane occupancy rates are low. During the undisturbed handling of a mega vessel it is recommend operating with less warehouse cranes than have been introduced on the reference terminal. In Chapter 8.5 more definite conclusions can be made as that chapter concerns the handling of 3 vessels moored at the terminal.

Rgv waiting queues on the nodes
The rgv row length on the nodes during adjacent loading has been plotted in Appendix 18. What can be concluded is that during that simulation run the row lengths on the nodes occasionally amounts to the length of 3 rgv's, but never 4 rgv's.

8.5 Simulation runs with 3 occupied berths

In the previous chapter the terminal has been tested during undisturbed conditions, with one moored vessel at the terminal. During this configuration no problems have been encountered. In this chapter the pressure on the terminal has been increased by having 3 vessels moored at the berths. Numerous combinations of loading and unloading strategies can be worked out, but only the 4 that cause the highest pressure on the terminal have been simulated.

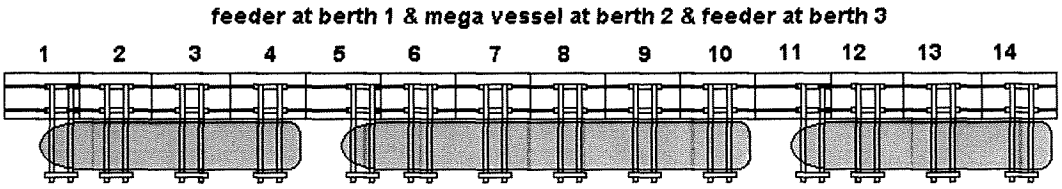


Figure 8.29, all the berths occupied

The next experiments have been carried out with all the berths occupied. The basic parameters used during these runs are shown in Appendix 20-23. The experiments are listed in the Table 8.5

Experiment	Berth 1	Berth 2	Berth 3
1.	Adjacent loading	Adjacent loading	Adjacent loading
2.	Adjacent loading	Unloading	Adjacent loading
3.	Unloading	Adjacent loading	Unloading
4.	Shuffling	Shuffling	Shuffling

Table 8.5, Experiments with 3 occupied berths

8.5.1 Experiment 1

This simulation run concerns the simultaneous adjacent loading of the three moored vessels. This situation may occur when all the berths are occupied and the pre-shuffling of containers has not taken place on the terminal. In this situation the nodes located on, or giving access to the inter-berth lanes will be tested at their maximum. The space that has to be reserved for waiting rows on the inter-berth lanes has been determined during this simulation run. For this reason also nodes located on the outer inter-berth lanes have been included in the output.

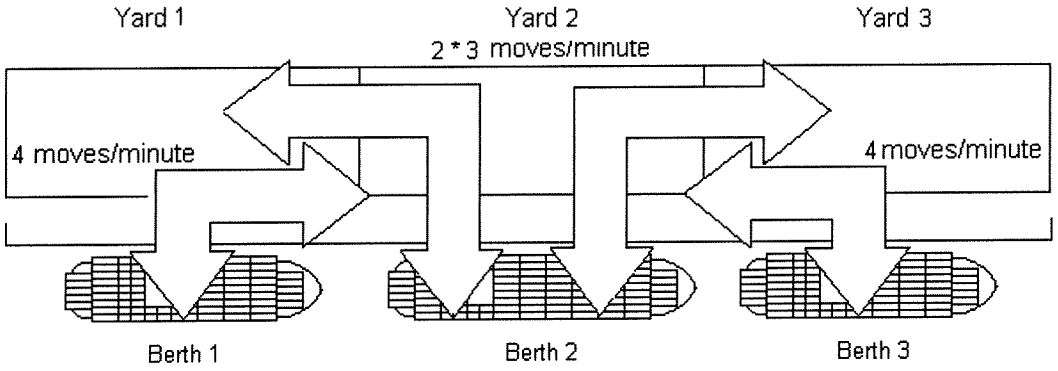


Figure 8.30, adjacent loading on all berths

Portainer occupancy rates
As can be seen in Figure 8.31, the occupancy rate of the portainers located on the feeder vessel berths, Portainer 1-4 and Portainer 11-12 are slightly lower than the portainer occupancy rates of the portainers located on Berth 2. The reason is that the rgv driving distances decrease from the Portainer 1 to Portainer 4 as the rgv's working for both portainers retrieve their containers from Yard 2. This also happens on Berth 3. As the rgv's working for the portainers located on Berth 2 retrieve their containers from both Yard 1 and Yard 2, the rgv driving distances are more constant, witch agrees with the figure and results in higher portainer occupancy rates.

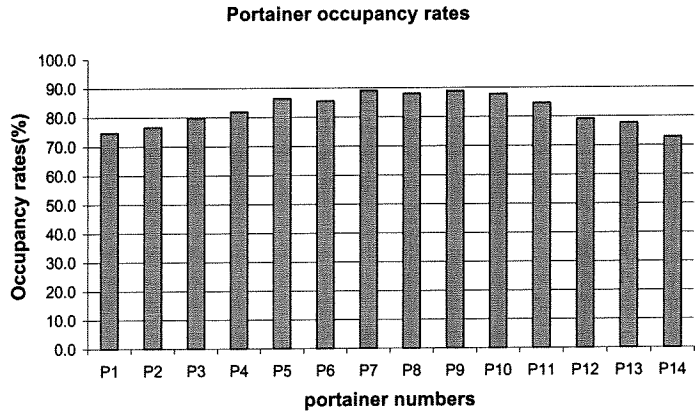


Figure 8.31, portainer occupancy rates

Handling production
The above described portainer occupancy rates result in productions per berth as shown in Table 8.4. These results agree with what might be suspected from the portainer occupancy rates. The mean portainer productions of Berth 1 and Berth 3 are lower than the mean portainer capacity of 60 moves per hour.

	Number of adjacent loaded containers	Production per portainer (moves/hour)	Production per portainer (moves/hour)	Production per berth (TEU/hour)	Service time (hours)
Berth 1	1050	48.6	194.3	291.6	5.4
Berth 2	3150	54.8	329.1	493.7	9.7
Berth 3	1050	48.6	195.5	293.3	5.4

Table 8.6, berth productions during adjacent loading on all berths

The handling times of the feeder vessels during adjacent loading on the terminal with three occupied berths is only 0.3 hour higher than the handling times during adjacent loading of only one moored feeder vessel.

This result can be defended by realising that the rgv's working for Berth 2 take the first inter-berth lane ring and that the rgv's working for Berth 1 & 3 take the second inter-berth lane ring thereby separating the inter-berth rgv flow. The fact that rgv's belonging to separate berths also retrieve their containers from separate yards means that the only place where the rgv flows will interact is at the entrances of the yard-lanes and on crossing points on the inter-berth lanes.

The fact that the rgv's working for Berth 1 & Berth 3 have to take the second ring means that they will always have to give the right of way to the rgv's working for Berth 2 that take the first ring. This is the reason that the service times on Berth 2 do not differ much from the undisturbed situation and that the service times on Berth 1 & 3 take 0.3 hours longer than the undisturbed situation.

The handling times of the mega vessels during adjacent loading on the terminal with three occupied berths is the more or less the same as the handling times during the undisturbed adjacent loading of only one moored mega vessel.

Rgv waiting queues under the portainers

The occupancy rates depend on the availability of rgv's waiting in the portainer queue. If the queue is empty the portainer will not be able to dispose of its container which is unfavourable for the portainer occupancy rate. A steady flow of rgv's is needed to keep the portainer occupancy rates high.

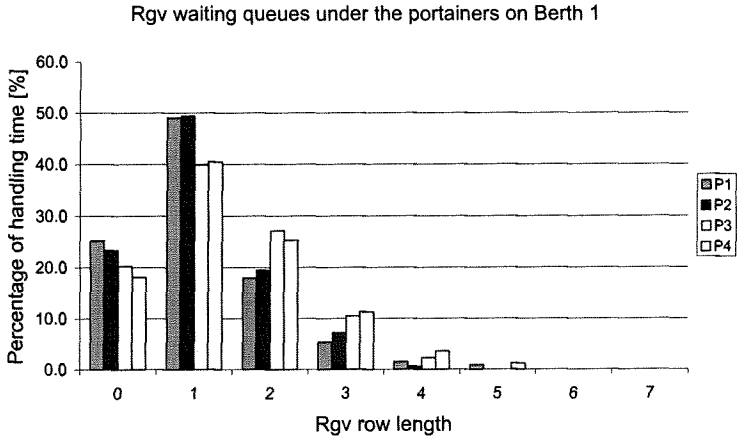


Figure 8.32, rgv waiting rows under the portainers on Berth 1

Under the portainers on Berth 1 the rgv waiting rows have not exceeded a number of 5 rgv's, as can be seen in Figure 8.32. Most of the time the waiting row under the portainers is 1 rgv. For about 20 % of the time no rgv's where present resulting in inactivity periods for the portainers. A solution could be to increase the number of rgv's operating for the portainers on Berth 1.

During the adjacent loading of the mega vessel moored at Berth 2, the rgv row has amounted to 6 rgv's only once. For the longest part of the time the rgv-waiting row has been 1 rgv. The rgv rows under the portainers where empty for about 12,5 % of the time, which is lower than the 20 % of Berth 1 and Berth 3. This is consistent with the fact that the production on Berth 2 is higher due to the shorter inactivity periods of the portainers. The rgv waiting rows under the portainers on Berth 2 can be seen in Figure 8.33.

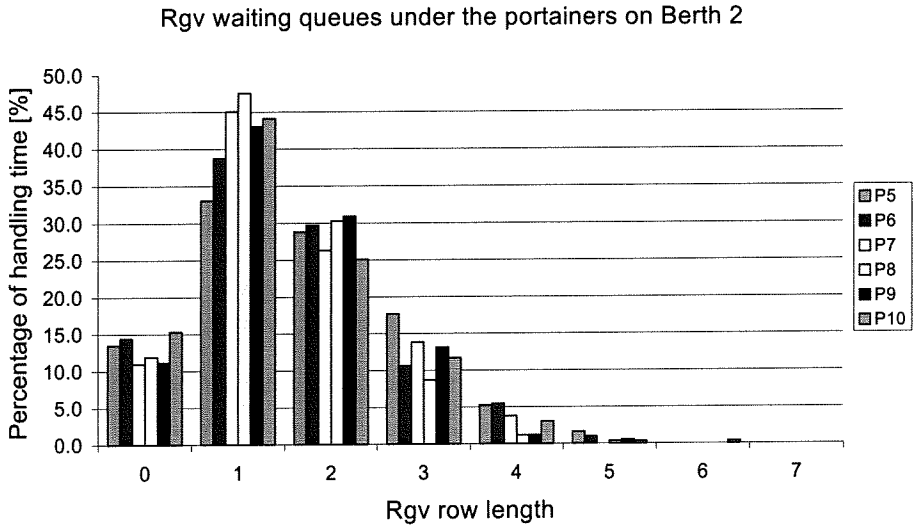


Figure 8.33, rgv waiting rows under the portainers on Berth 2

The rgv waiting rows under the portainers located on Berth 3 are comparable to the waiting rows on Berth 1. This can be seen in Figure 8.34.

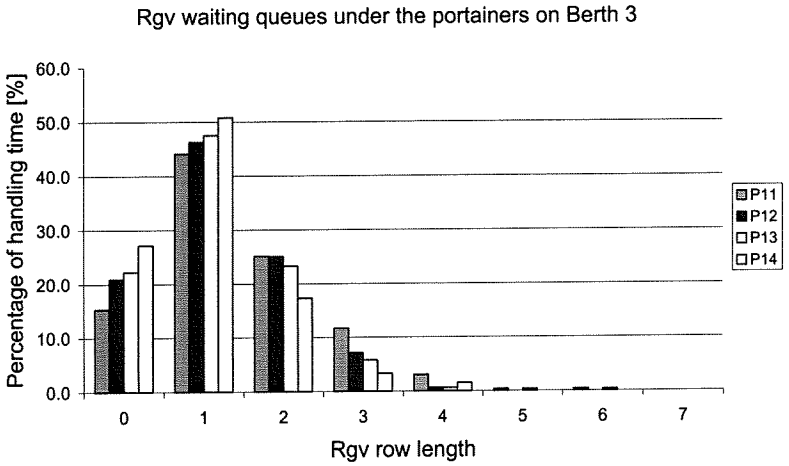


Figure 8.34, rgv waiting rows under the portainers on Berth 3

Rgv waiting queues at the nodes

The nodes with the expected highest rgv rows have been analysed in order to get an impression on the busyness of the terminal during the adjacent loading of three vessels. In Appendix 7 the rgv flows on the terminal have been calculated for this situation. From that calculation the nodes most liable to suffer long rgv rows have been identified. These are the nodes 110, 350, 108, 380 and 106 that connect the top inter-berth lanes with the yard-lanes. The nodes 508, 333, 510, 363 and 512 connect the bottom inter-berth lanes with the yard-lanes. A number of nodes located on more outward situated inter-berth lanes have been included in order to determine the needed space for rgv waiting rows. The waiting rows of these nodes have been plotted in Appendix 20. What can be seen is that only the length of the rgv queue on node 350 reaches 6 rgv's once. The maximum rgv length on the nodes does not exceed the length of 4 except on node 350.

Rgv waiting rows on the warehouse lanes
The warehouse lanes of the warehouses in Yard 2 are the busiest on the terminal for this situation. The stack-cluster located behind portainer location 11, Stacks 101-110, has been included in Appendix 20 to give an impression of the rgv row length on the different lanes. Apart from some initial effects the rgv row only once exceeds the length of 2 rgv's. As rgv row of three are acceptable the stacks and three warehouse cranes seem to suffice.

Warehouse crane occupancy rates
The warehouse crane occupancy rates for the warehouse cranes located in Yard 1, 2 and 3 have been calculated for the simultaneous adjacent loading on all three berths. The occupancy rates of these warehouse cranes have been calculated in Appendix 20, resulting in:

Mean occupancy rate for a warehouse crane located in the 20' stack is 34,7 %.
Mean occupancy rate for a warehouse crane located in the 40' stack is 23,2 %.

Mean occupancy rate for a warehouse crane located in the 20' stack is 43,4 %.
Mean occupancy rate for a warehouse crane located in the 40' stack is 29,0 %.

These occupancy rates seem acceptable. It might even be possible to operate with fewer warehouse cranes.

8.5.2 Experiment 2

This simulation run concerns the adjacent loading of two feeder vessels moored at Berth 1 & 3 and the unloading of a mega vessel moored at Berth 2. This situation has been analysed in Appendix V.6 and has turned out to be the heaviest occupied situation for the lane, node and warehouse crane capacities on Yard 2. This is the reason that this situation has been chosen to be the normative situation on the terminal. In Chapter 8.6 some variations on the reference terminal will be carried out with this terminal configuration.

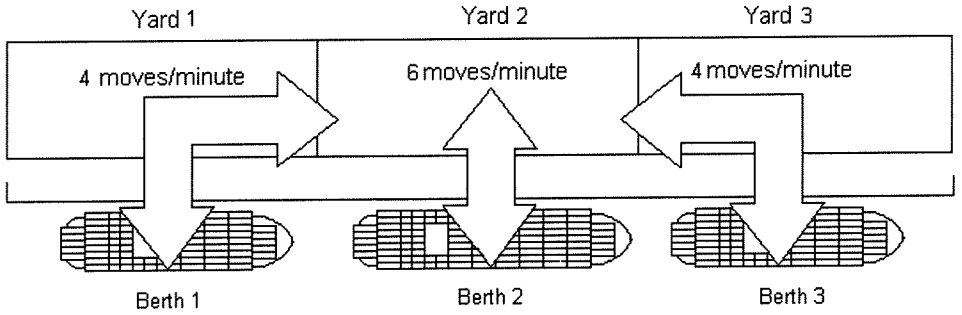


Figure 8.35, adjacent loading on Berth 1 & 3 and unloading on Berth 2

Portainer occupancy rates
The occupancy rates of the portainers located on Berth 1 & 3 are comparable to the occupancy rates found in Experiment I of this chapter. The portainer occupancy rates become lower as the portainer is situated further away from Berth 2.
The portainer occupancy rates of the portainers located on Berth 2 show a similar picture as during the unloading without interference; Experiment I of Chapter 8.4. The occupancy rates are not as high, but the differences in odd- and even-numbered portainers are apparent again.

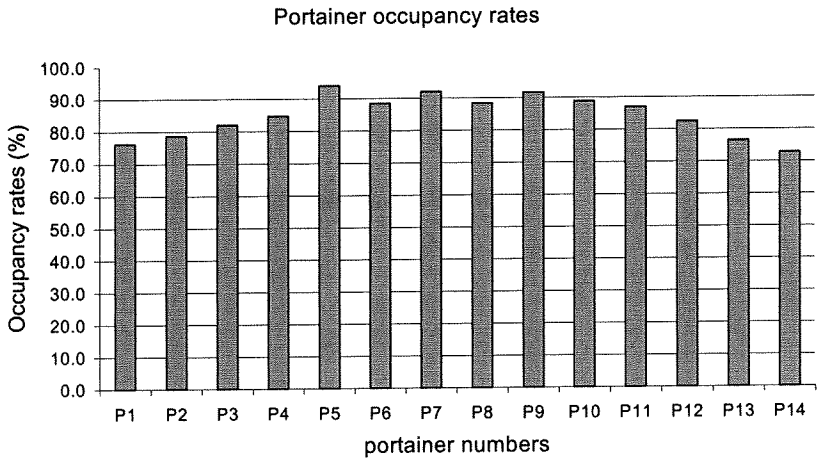


Figure 8.36, Portainer occupancy rates

Handling production
The portainer occupancy rates as shown in Figure 8.36 result in the productions per berth and the vessel service times as shown in Table 8.5.

	Number of handled containers	Production per portainer (moves/hour)	Production per berth (moves/hour)	Production per berth (TEU/hour)	Service time (hours)
Berth 1	1050 (adj. load)	49.9	199.6	299.3	5.3
Berth 2	3150 (unload)	56.2	343.9	515.8	9.3
Berth 3	1050 (adj. load)	49.4	197.8	296.7	5.3

Table 8.7, berth productions

The service times for both adjacent loading feeder vessels are 0.2 hours longer compared to the undisturbed adjacent loading feeder vessel. The service time of the mega vessel is also 0.4 hours slower than the service times found during the undisturbed unloading.

Rgv waiting queues under the portainers
The length of the portainer queues is related to the portainer occupancy rates. The bigger the share of an empty queue in Figure 8.37 the lower the occupancy rates will be.

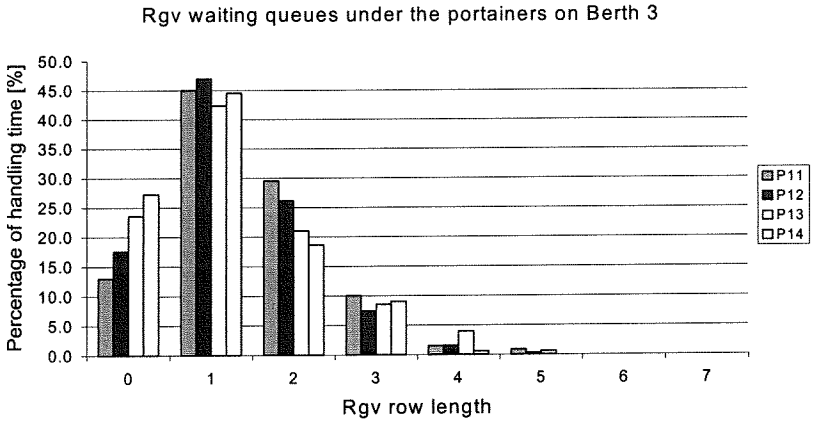


Figure 8.37, rgv waiting rows under the portainers on Berth 1

Under the portainers of Berth 1 the rgv rows amount to 6 in length only once. This happens during the adjacent loading on Berth 1. The share of the empty rgv rows is about 20 %. Increasing the numbers of rgv's on the terminal during adjacent loading to 12 might give more favourable results.

During the unloading of the mega vessel on Berth 2, the waiting row under the portainers occasionally grows to 6 rgv's in length. The share of an empty rgv row is about 7 %.

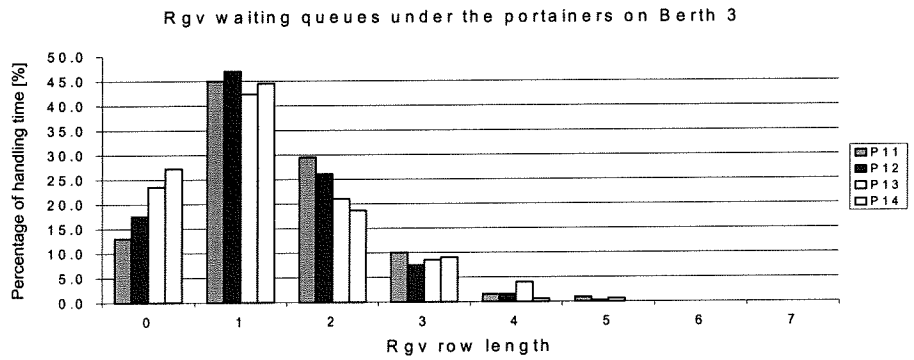


Figure 8.38, rgv waiting rows under the portainers on Berth 2

The rgv waiting rows under the portainers located on Berth 3 are comparable to the waiting rows on Berth 1. This can be seen in Figure 8.39.

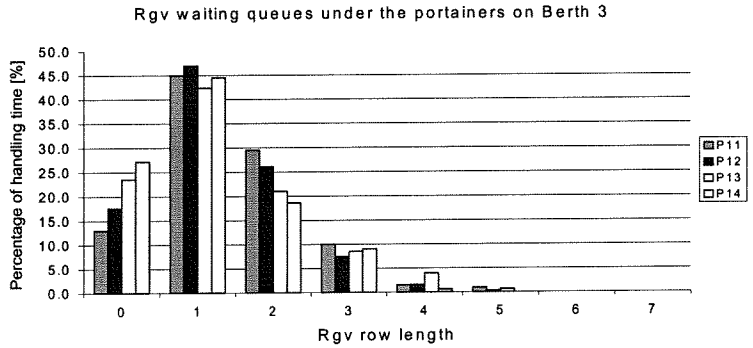


Figure 8.39, rgv waiting rows under the portainers on Berth 3

Rgv waiting queues on the nodes

In Appendix 7 the rgv flows on the terminal have been calculated for this situation. From that calculation the nodes most liable to suffer long rgv rows have been identified. These are the nodes 110, 350, 108, 380 and 106 that connect the top inter-berth lanes with the yard-lanes. The nodes 508, 333, 510, 363 and 512 connect the bottom inter-berth lanes with the yard-lanes. The waiting rows of these nodes have been plotted in Appendix 21. What can be seen is the rgv waiting queues on the node seldom exceeds the length of 2 rgv's.

Rgv waiting rows on the warehouse lanes

The warehouse lanes of the warehouses located in Yard 2 are the busiest on the terminal for this situation. The stack cluster located behind portainer 11, Stacks 101-110, has been included in Appendix 21 to give an impression of the rgv row length on the different lanes. The rgv row on Warehouse lane 104 is located in a 20' stack. The occupancy is high and at one point even exceeds the maximum length of 3 rgv's as can be seen in Figure 8.40.

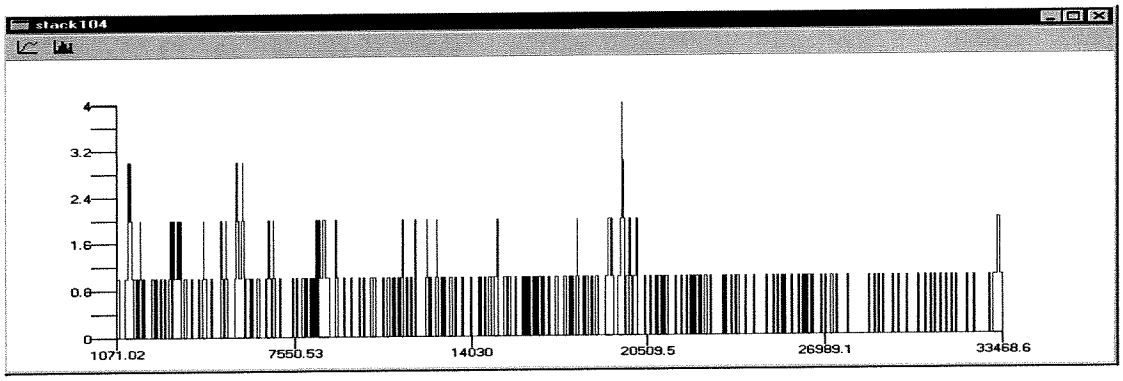


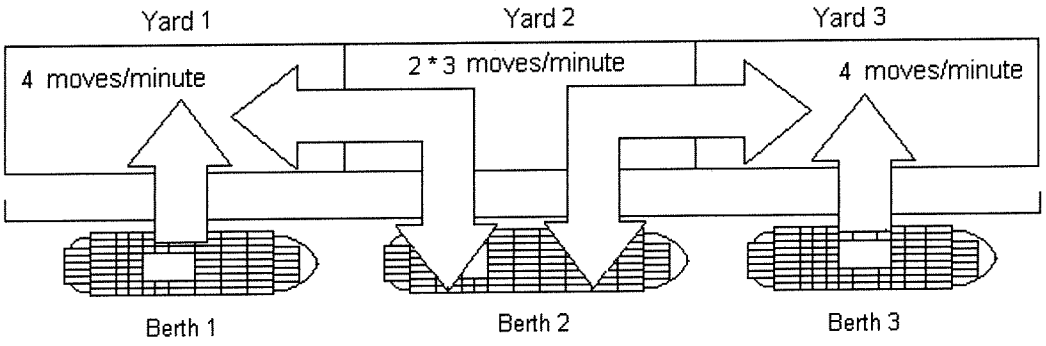
Figure 8.40, rgv row on Warehouse lane 104

Warehouse crane occupancy rates
The warehouse crane occupancy rates for the warehouse cranes located in Yard 2 are the highest on the terminal during the simultaneous adjacent loading on all three berths. The occupancy rates of these warehouse cranes have been calculated in Appendix 21, resulting in:

Mean occupancy rate for a warehouse crane located in the 20' stack is 70.1 %.
Mean occupancy rate for a warehouse crane located in the 40' stack is 47,3 %.

These occupancy rates are reasonably high which can also be concluded from the rgv rows on the warehouse lanes.

8.5.3 Experiment 3



This simulation run concerns the unloading of the two feeder vessels moored at Berth 1 & 3 and the adjacent loading of a mega vessel moored at Berth 2. This configuration provides the highest occupancy of the yards situated behind the feeder vessel berths.

Portainer occupancy rates
The portainer occupancy rates of both feeder vessels are quite recognisable. The portainer occupancy rates of the odd numbered portainers are slightly higher than those of the even numbered portainers. This is due to the relative longer driving cycles of the even numbered portainers. The portainers located on Berth 2 have a mean occupancy rate of about 83%. This can be increased by assigning more rgv's to these portainers.

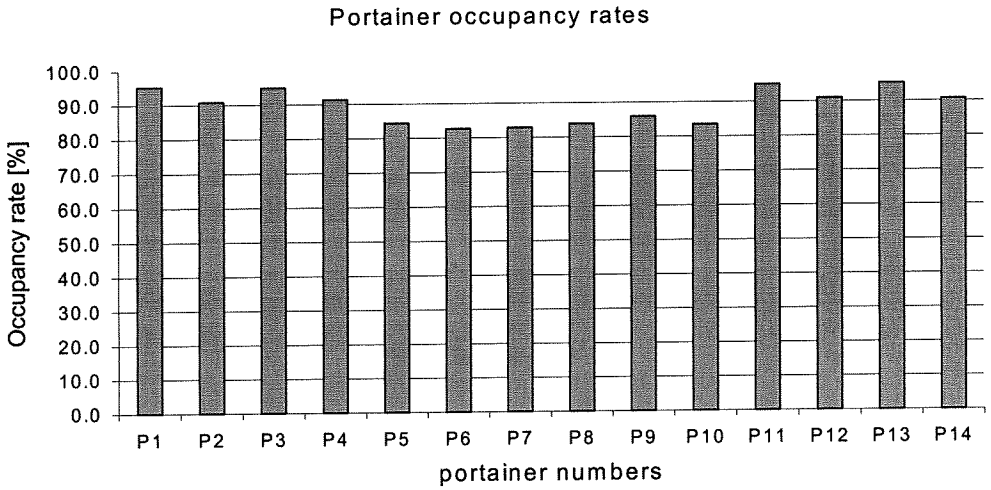


Figure 8.41, portainer occupancy rates

Handling production
Table 8.6 shows the handling production that can be achieved with this configuration. Despite the fact that the rgv's from Berth 2 retrieve their containers from Yard 1 & 3, both feeder vessel service times are not influenced. The undisturbed service time for the unloading of a feeder vessel also equals 4.5 hours.

	Number of handled containers	Production per portainer (moves/hour)	Production per berth (moves/hour)	Production per berth (TEU/hour)	Service time (hours)
Berth 1	1050 (unload)	57.8	231.4	347.1	4.5
Berth 2	3150 (adj. load)	50.1	300.8	451.2	10.5
Berth 3	1050 (unload)	57.7	231.0	346.5	4.5

Table 8.8, berth productions

The adjacent loading of a mega vessel with containers being retrieved from Yard 1 & 3 takes about 10.5 hours. This is 0.8 hours more that the undisturbed adjacent loading of a mega vessel which takes 9.7 hours.

Rgv waiting queues under the portainers
The rgv rows located under the portainers on Berth 1 have not exceeded the length of 4 rgv's during this simulation run. The small share of empty rgv rows under the portainers agrees with the high occupancy rates of the portainers.

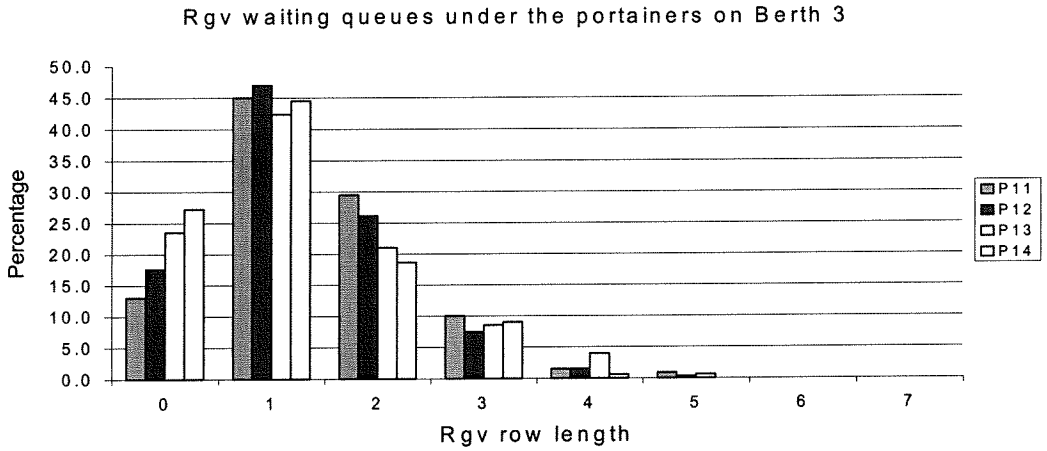


Figure 8.42, rgv waiting rows under the portainers on Berth 1

Figure 8.43 illustrates the length of the rgv waiting rows under the portainers located under the portainers on Berth 2. The empty row share is considerably higher compared to the rgv rows on Berth 1. The length of the rgv waiting row has amounted to 6 during this simulation run.

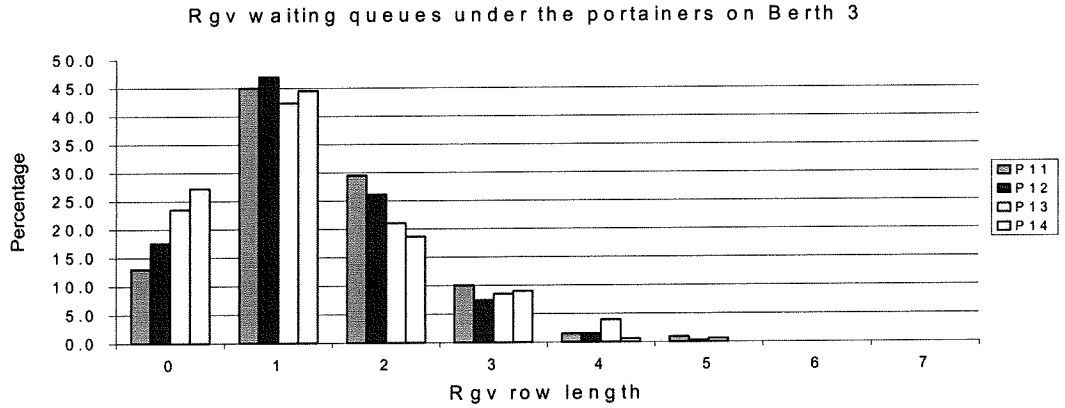


Figure 8.43, rgv waiting rows under the portainers on Berth 2

Rgv waiting rows at the nodes
The nodes most liable to have substantial rgv waiting rows are the access and exit nodes to the yard-lanes of Berth 1 & 3. The are the nodes 114, 243, 260, 273, 290, 504 and 506. These nodes have been joined to Appendix 22. The rgv rows on these nodes have never reached the length of 4 rgv's during this simulation run.

Rgv waiting rows on the warehouse lanes
The warehouse lanes of the warehouses located in Yards 1 & 3 that are used by the unloading rgv's have the highest occupancy. The stack-cluster containing the warehouse lanes 31-40 has been joined to Appendix 22. None of the rgv rows on the warehouse lanes on Berth 1 & 3 has exceeded the maximum length of 3, except warehouse lane 33.

Warehouse crane occupancy rates
The warehouse crane occupancy rates for the warehouse cranes located in Yard 1 & Yard 3 are the highest during this simulation run. They have been calculated in Appendix 22.

Mean occupancy rate for a warehouse crane located in the 20' stack is 65.6 %.
Mean occupancy rate for a warehouse crane located in the 40' stack is 43,7 %.

These occupancy rates are reasonably high which can also be concluded from the rgv rows on the warehouse lanes.

8.5.4 Experiment 4

In this experiment all the rgv's will be shuffling containers from their 'home' berth to an adjacent berth. This situation may occur when container vessels, having unloaded their containers in the yard behind their quays, have just left the terminal. During shuffling the rgv's are not subscribed to the portainers anymore but to the stack-clusters located behind the portainers. Per stack-cluster 10 rgv's will be active. Considering that shuffling will take place on all three the berths 10 * 14 stack-clusters = 140 rgv's will be active. This is the same amount of rgv's as during adjacent loading on three berths. As the portainers are not active during shuffling they will not need to be analysed.

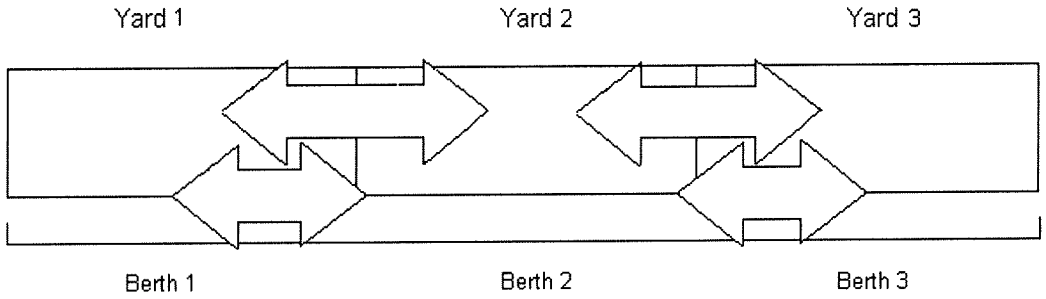


Figure 8.44, shuffling from all berths

Shuffle production
The shuffle production of the terminal during the shuffling on three berths will be defined as the total amount of shuffled containers per time unit. The total amount of shuffled containers is:

- Number of shuffled containers from Yard 1 to Yard 2: 1200
- Number of shuffled containers from Yard 2 to Yard 1: 1600
- Number of shuffled containers from Yard 2 to Yard 3: 1600
- Number of shuffled containers from Yard 3 to Yard 1: 1200

Total amount of shuffled containers on the terminal: 5600

The time necessary to shuffle this amount of containers is 23400 seconds or 6.5 hours.

This results in a total shuffle production of: $5{,}600 / 6.5 = 861.5$ containers per hour, or 14.4 containers per minute.

Rgv waiting queues at the nodes
The nodes with the expected highest rgv rows have been analysed in order to get an impression on the busyness on the terminal during the shuffling. These nodes are the same ones as have been analysed during the adjacent loading on all three berths; Experiment 1.
The nodes with the highest expected rgv rows are nodes 110, 350, 108, 106 and 508, 333, 510, 363 and 512. The waiting rows of these nodes have been plotted in Appendix 23. What can be seen is that the lengths of the rgv queues occasionally exceed the length of 3 rgv's, but never the length of 4 rgv's. The node with the longest rgv queue is node 380 as can be seen in Figure 8.45.

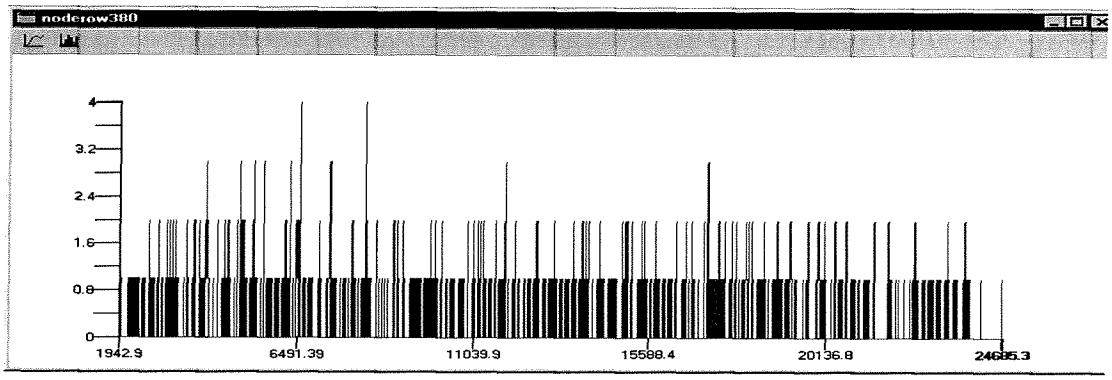


Figure 8.45, rgv waiting queue on node 380

Waiting rows in the warehouse lanes
The warehouse lanes of the warehouses in Yard 2 are the ones that will be visited most frequently during the simulation. Due to the high occupancy rates of the warehouse cranes the rgv queues on the warehouse lanes may become to long. As an example the rgv queue on warehouse lane 101 has been plotted in Figure 8.46.

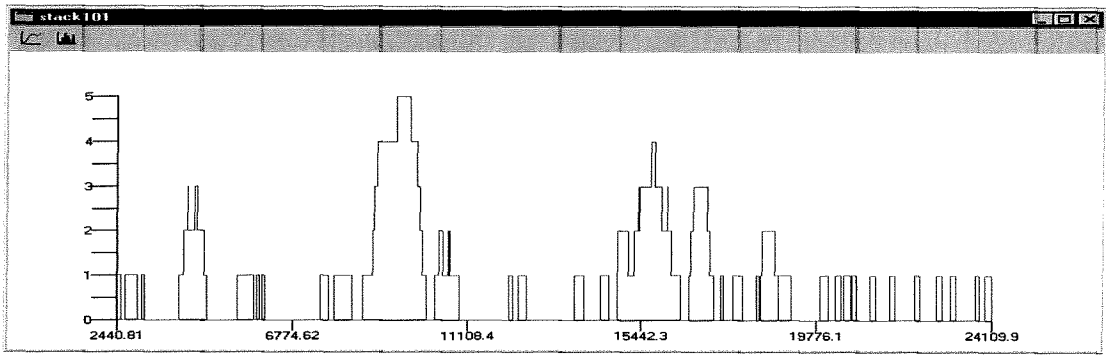


Figure 8.46, rgv row on warehouse lane 101

Warehouse crane occupancy rates
The warehouse crane occupancy rates for the warehouse cranes located in Yard 2 are the highest on the terminal during shuffling on all three the berths. The occupancy rates of these warehouse cranes have been calculated in Appendix 23, resulting in:

Mean occupancy rate for a warehouse crane located in the 20' stack in Yard 2 is 79.8 %
Mean occupancy rate for a warehouse crane located in the 20' stack in Yard 2 is 53.2 %

These occupancy rates are extremely high, especially those of the 20' stacks and its warehouse cranes. This, of course, is the main cause that the lengths of the waiting rows on the warehouse lanes are large. The reason that the warehouse crane occupancy rates are this high is because the numbers of shuffling rgv's per stack-cluster is too high. By decreasing the number of shuffle rgv's per stack-cluster the occupancy rate of the warehouse cranes and the resulting rgv waiting rows will become less. The time needed to shuffle the same amount of containers will increase by appointing less shuffle rgv's.

Bearing in mind the relative small differences between the adjacent loading and the unloading of a feeder or mega vessel and the extra amount of moves needed to shuffle the containers to the yards from where they can be fast loaded, shuffling will not be that profitable. The main reason is that the driving distances between the Yards are not far enough. Shuffling will therefore not be recommended.

8.5.5 Recapitulation on the simulation runs with 3 occupied berths

Handling rates
The service times performances found in Experiment 1-3 during the simultaneous handling of 3 container vessels have been plotted in Table 8.9.

	Feeder vessel moored at Berth 1 or 3		Mega vessel moored at Berth 2	
	Unloading [hours]	Adjacent loading [hours]	Unloading [hours]	Adjacent loading [hours]
Experiment 1	--	5.4	--	9.7
Experiment 2	--	5.3	9.3	--
Experiment 3	4.5	--	--	10.5

Table 8.9, handling rates during simulation on 3 berths

From the handling times displayed in Table 8.9, the service times of a feeder vessel and a mega vessel during high occupation on the reference terminal can be established. For a feeder vessel the maximum service time with 3 vessels moored vessels at the terminal is 5.4 + 4.5 = 9.9 hours. For a mega vessel moored at Berth 2 the same approach can be made. The maximum service time on the reference terminal will amount to 9.3 + 10.5 = 19.8 hours.

In Table 8.10 a comparison is made between the undisturbed service times and the maximum service times on the reference terminal.

	Undisturbed service times [hours]			Maximum service times [hours]			Difference [hours]
	Unloading	Adjacent loading	Total	Unloading	Adjacent loading	Total	
Feeder vessel	4.5	5.1	9.6	4.5	5.4	9.9	0.3
Mega vessel	8.9	9.7	18.6	9.2	10.5	19.8	1.2

Table 8.10, comparison between the undisturbed and maximum service times

What can be interpreted from Table 8.10 is that the differences between the undisturbed service times and the maximum service times on the reference terminal are not excessive. For a feeder vessel the difference as a percentage of the maximum service times is 3.3 %, while for the mega vessel it is 6.1 %.

These results defend the proposition to exclude the shuffling from the handling procedures on the terminal. The most important reason that the shuffling is not practicable for the reference terminal is the fact that the driving distances between the 3 yards is relatively small due to the compactness of the terminal. Retrieving a container from an adjacent stack during the loading of a vessel will not require immense driving distances compared to conventional terminals. Another very important issue is the fact that the containers are stacked in warehouses. This means that the amounts of shuffling moves in the yard itself are also strongly reduced. On a conventional terminal container stacks are pre-shuffled in such a way that the first containers to be loaded onto the vessel are easy reachable.

Operational rgv's - portainer occupancies
The amount of operational rgv's on the reference terminal during the simulation runs with 3 occupied berths has been established to be 6 rgv's per portainer during the unloading, and 10 per portainer during the adjacent loading of a container vessel. As the portainer occupancy rates during the adjacent loading of a feeder or mega vessel are relatively low increasing the number of rgv's to 12 per portainer might be a solution. Especially considering the fact that the waiting rows on the nodes and warehouse lanes are not very long. A simulation run with 12 rgv's has been carried out in the next chapter. The simulation run is based on the situation as it exists in Experiment 2.

Warehouse crane occupancy rates
The warehouse crane occupancy rates on the reference terminal with 3 occupied berths have been included to Table 8.11.

	Warehouse crane occupancy rates with 3 occupied berths (%)					
	Yard 1		Yard 2		Yard 3	
	20' stack	40' stack	20' stack	40' stack	20' stack	40' stack
Experiment 1	43.4	29.0	34.7	23.2	43.4	29.0
Experiment 2	--	--	70.1	47.3	--	--
Experiment 3	65.6	43.7	--	--	65.6	43.7

Table 8.11, warehouse crane occupancy rates

With the data from this table the necessary number of warehouse cranes can be determined. This has been done below.

First the normative situation must be determined from Experiment 1-3 by establishing the mean warehouse crane occupancy rate over all the yards.

The 20' container stacks:

Exp. 1: $\frac{43.4*4+34.7*6+43.4*4}{14}=40\%$

Exp. 2: $\frac{0*4+70.1*6+0*4}{14}=30\%$

Exp. 3: $\frac{65.6*4+0*6+65.6*4}{14}=37.5\%$

The occupancy rate is the normative for Experiment 1. The warehouse cranes in the 20' stacks have a mean occupancy rate of 40%. The target occupancy rate for this situation is 70 %, resulting in 16 warehouse cranes in the 20' stacks.

The same calculation has been made for the 40' stack:

Exp. 1: $\frac{29.0*4+23.2*6+29.0*4}{14}=26.5\%$

Exp. 2: $\frac{0*4+6*47.3+0*4}{14}=20.3\%$

Exp. 3: $$\frac{43.7*4+0*6+43.7*4}{14}=25\%$$

The occupancy rate is the normative for Experiment 1. The warehouse cranes in the 40' stacks have a mean occupancy rate of 26.5%. The target occupancy rate for this situation is 70 %, resulting in 16 warehouse cranes in the 40' stacks. As there are three warehouse rows present in the 40' yards the number of warehouse cranes has been determined to amount to 18.

It has to be noted that during all simulation runs on the reference terminal the warehouse crane handling cycles have not been variable but deterministic. Variable operating warehouse cranes will probably have slightly different occupancy rates. For this reason a simulation run has been carried out with variable handling cycles. This simulation run is based on the situation in Experiment 2 as it has the highest occupation rates in the yard.

Rgv waiting queues on the nodes
The normative situation for determining the needed rgv waiting row space between 2 inter-berth lanes is the adjacent loading on all 3 berths. In this situation the inter-berth lanes will have to digest the largest amount of rgv's. The required space for these waiting rows has been based on the row lengths located between the crossings of inter-berth lanes with the extensions of the yard-lanes. Figure 8.47 provides an illustration where the locations of the waiting places are represented by the black dots.

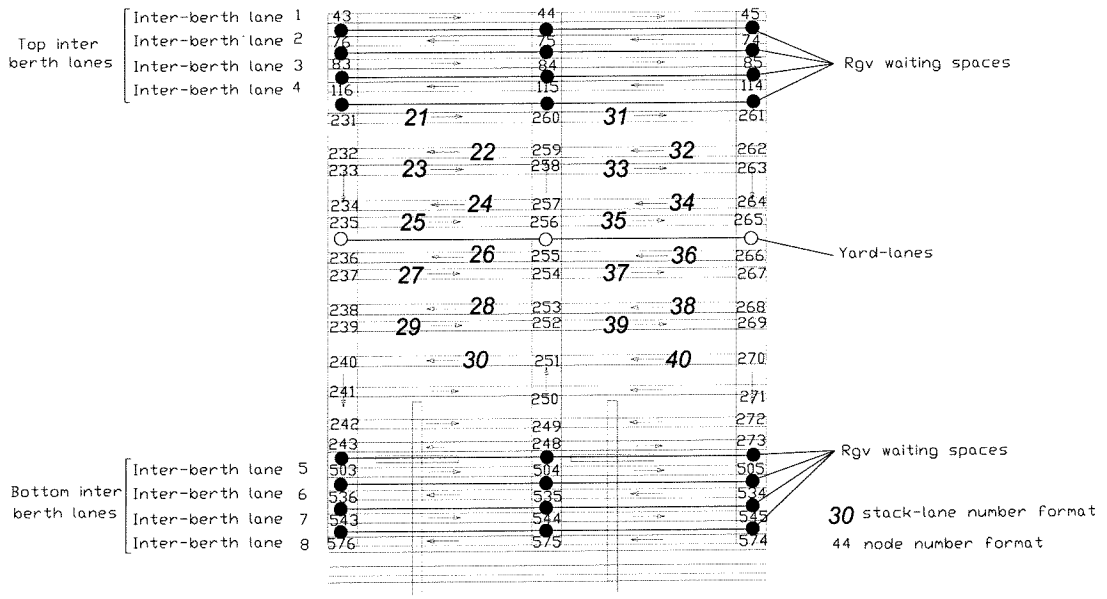


Figure 8.47, a part of the top-view of the layout of the reference terminal

The simulation run has led to the following maximum row lengths.

Top inter-berth lanes.
- Maximum row length that has occurred between inter-berth lane 1 and 2: 3 rgv's
- Maximum row length that has occurred between inter-berth lane 2 and 3: 3 rgv's
- Maximum row length that has occurred between inter-berth lane 3 and 4: 5 rgv's
- Maximum row length that has occurred between inter-berth lane 4 and yard-lanes: 5 rgv's

Bottom inter-berth lanes.
- Maximum row length that has occurred between inter-berth lane 8 and 7: 3 rgv's
- Maximum row length that has occurred between inter-berth lane 7 and 6: 4 rgv's

- Maximum row length that has occurred between inter-berth lane 6 and 5: 4 rgv's
- Maximum row length that has occurred between inter-berth lane 5 and yard-lanes: 4 rgv's

These maximum row lengths will occasionally occur during adjacent loading on all 3 berths. In the next chapter these results will be discussed and integrated in the layout of the terminal.

8.6 Simulation runs of variations on the reference terminal

In the last chapters a variety of runs have been carried out based on the reference terminal. In this chapter some alterations to the reference terminal have been simulated and analysed. These alternations have been applied to the normative situation on the terminal. The established normative situation is the adjacent loading of the feeder vessels moored at Berth 1 & 3 combined with the unloading of the mega vessel moored at Berth 2. This is Experiment 2 of Chapter 8.5.

The experiments and their alternations to the reference terminal can be seen in Table 8.12.

Experiment	Berth 1	Berth 2	Berth 3	Alternation
1.	Adjacent loading	Unloading	Adjacent loading	From 10 to 12 rgv's per portainer on the adjacent loading berths
2.	Adjacent loading	Unloading	Adjacent loading	The addition of disturbances to the portainer handling cycles
3.	Adjacent loading	Unloading	Adjacent loading	Crane handling times have been made variable instead of deterministic

Table 8.12, description of alterations to the reference terminal

8.6.1 Increasing the number of adjacent loading rgv's to 12

During this simulation run the number of active rgv's working for the portainers located on the feeder vessel Berths 1 & 3 have been increased to 12. The most important results of this alternation will be dealt with in this chapter.

Portainer occupancy rates
As can be seen in Figure 8.47 the portainer occupancy rates of the portainers located on Berth 1 & 3 have increased with about 10 % compared to the reference terminal.

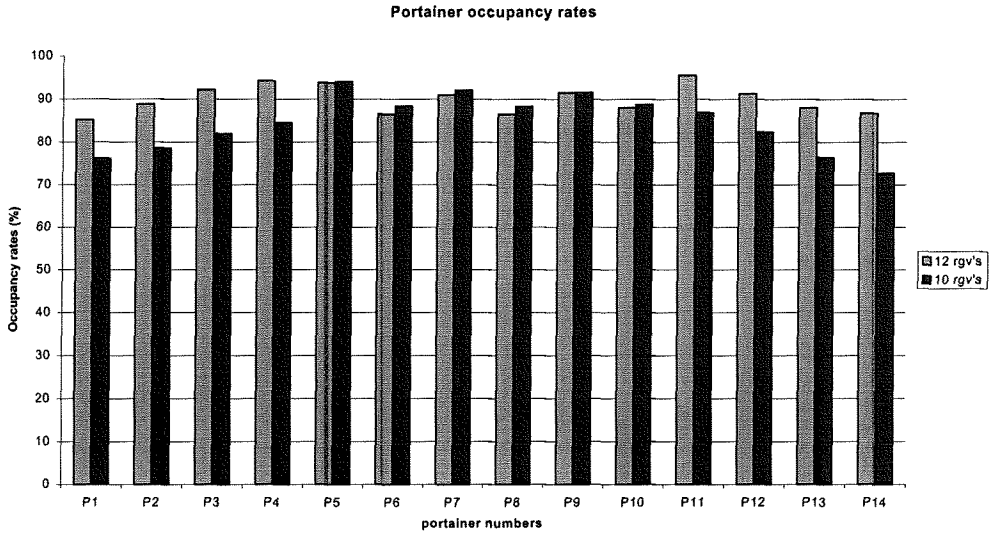


Figure 8.48, portainer occupancy rates

Handling production
The handling production on the three berths can be seen in Table 8.13.

	Number of adjacent loaded containers	Production per portainer (moves/hour)	Production per berth (moves/hour)	Production per berth (TEU/hour)	Service times (hours)
Berth 1	1050 (adj. load)	55.9	223.8	335.7	4.7
Berth 2	3150 (unload)	54.8	328.8	493.2	9.6
Berth 3	1050 (adj. load)	56.1	224.5	336.8	4.7

Table 8.13, handling productions with 12 rgv's

Compared to the similar run with 10 rgv's the service times on Berth 1 & 3 have decreased to under the service times of the undisturbed adjacent loading of a feeder vessel. The service times no Berth 2 have increased a bit. The increase of the service time on Berth 2 is due to the fact that that there are more rgv's on the lanes and in the stacks causing some delay.

Rgv waiting queues under the portainers

Compared to the similar run with 10 rgv's, the percentage of empty waiting rows under the portainers on Berth 1 & 3 have become smaller. This can be seen in the following figures.

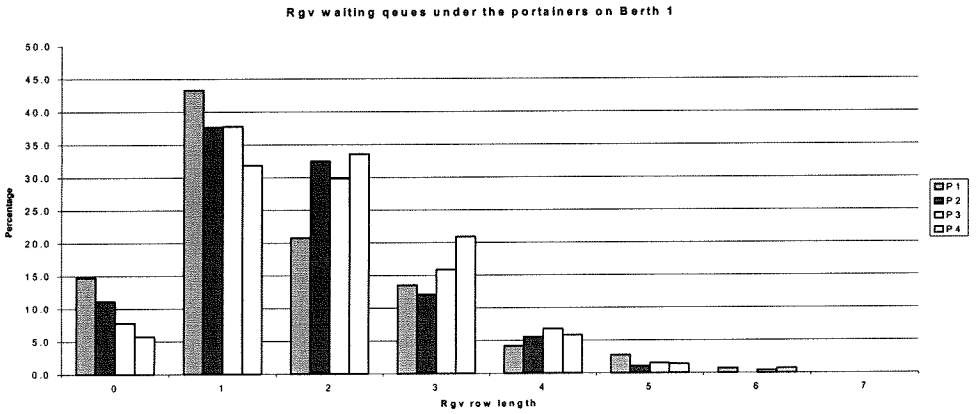


figure 8.49, rgv waiting rows under the portainers on Berth 1

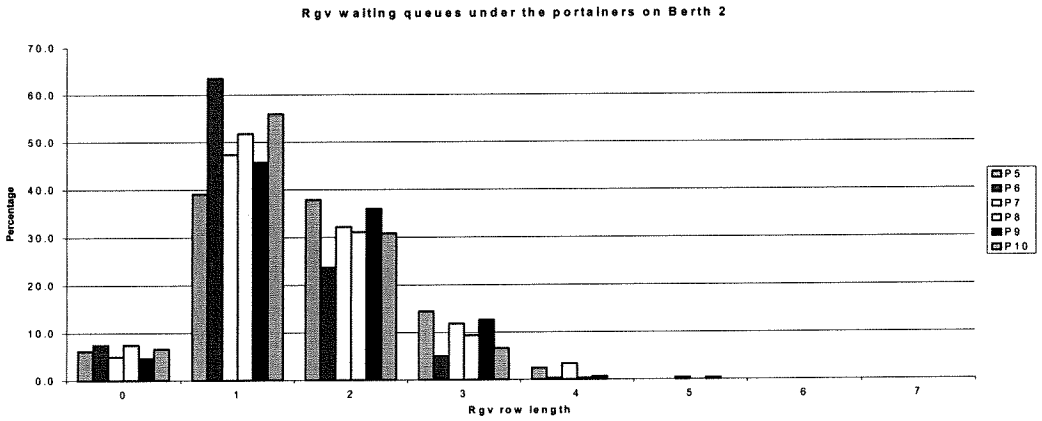


Figure 8.50, rgv waiting rows under the portainers on Berth 2

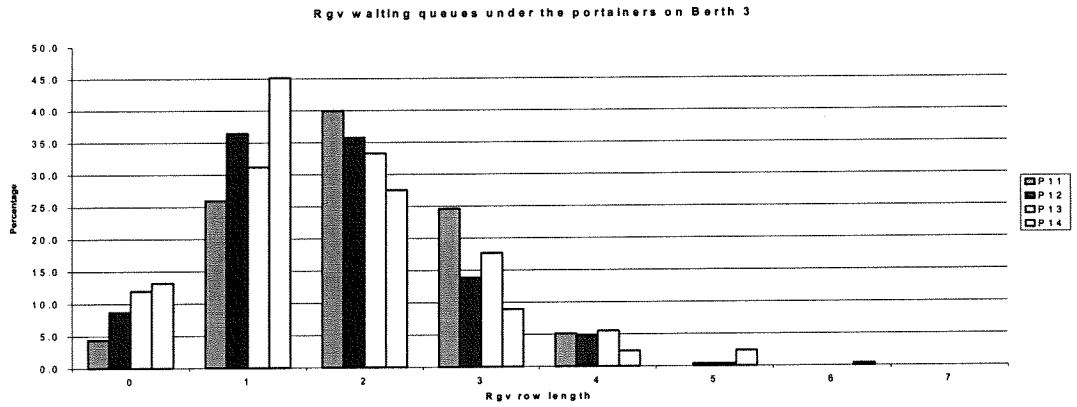


Figure 8.51,, rgv waiting rows under the portainers on Berth 3

Rgv waiting queues on the nodes and on the warehouse lanes
In Appendix VIII.13 the rgv waiting queues on the nodes and warehouse lanes can be found.
They pose no threats to the terminal

8.6.2 The loading and unloading of a vessel using different portainer cycle times

Experiment
The portainer cycle times used at the reference terminal are the shortest cycle times possible. In reality these cycle times will not be this perfect. Because that would mean the portainer driver will not make any mistakes that would decrease the cycle times of the portainer. Therefore some extra time penalties have been added to model to be interpreted as disturbances. Two kinds of disturbances are added.

A small disturbance with a maximum of 5 seconds and a distribution function as shown in Figure 8.52.

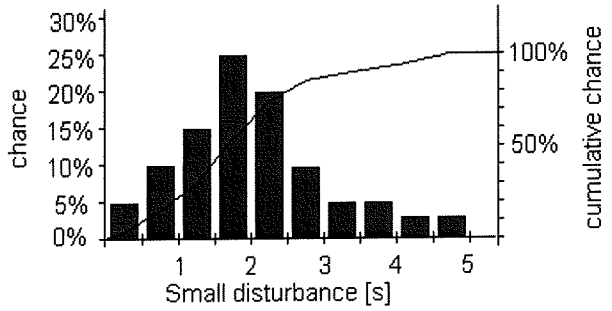


Figure 8.52, distribution of a small disturbance in the portainer handling time

A big disturbance that only occurs once in a hundred portainer moves, with a mean of 300 [s] as shown in Figure 8.53.

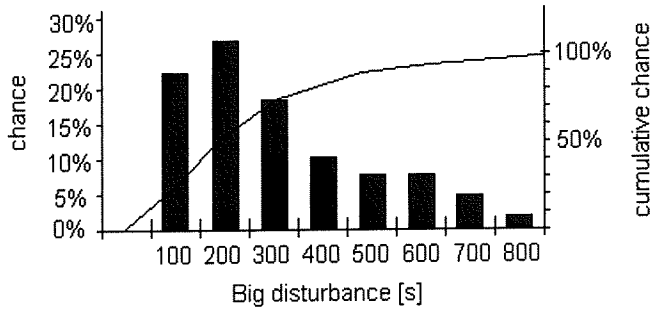


Figure 8.53, distribution of a big disturbance in the portainer handling time

Including these two disturbances, the new mean portainer cycle time is 63 seconds.

The portainer cycle times

The mean portainer cycle times of the basic run are shown in Table 8.14 where they are compared with the cycle times of the experiment.

Portainer	Old portainer cycle time [s]	experiment portainer cycle time [s]
5	56.66	59.28
6	57.04	61.45
7	56.90	60.58
8	57.00	62.54
9	56.86	61.08
10	57.08	65.85
mean cycle time	**56.92**	**61.80**

Table 8.14, the mean portainer cycle times

In this table the difference between the old and the new mean cycle times becomes clear. The difference is 4.88 seconds.

Using the cycle times and the occupancy rates of the portainers, the productions per portainer can be calculated. It is clear that the portainer productions will be lower in the experiment while the mean cycle times are higher.

First the portainer production is considered without the portainer occupancy rates, the production is defined as in (8.1)

$$\text{production}[\text{moves}/\text{min}] = 60/\ \text{mean cycle time}[\text{sec}/\text{move}] \qquad (8.1)$$

The mean cycle times of the two runs as shown in table could suggest that the portainer productions in the experiment are 8.6 % lower than in the original situation (8.2).

$$\frac{61.8[\text{sec}]}{56.9[\text{sec}]} = 1.086 \rightarrow 8.6\% \qquad (8.2)$$

Figure 8.53 shows the portainer productions from the run compared to the original situation. Relatively the productions are not as low as expected. This is because the portainer occupancy rates in the experiment are higher than in the original run.

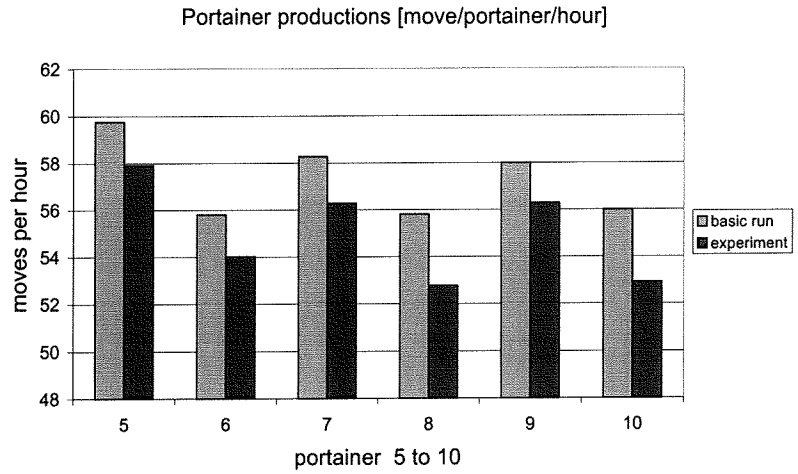


Figure 8.54, the portainer productions at berth 2

The expected portainer productions on basis of the cycle times, the result of the basic run and the result of the experiment are shown in Figure 8.55.

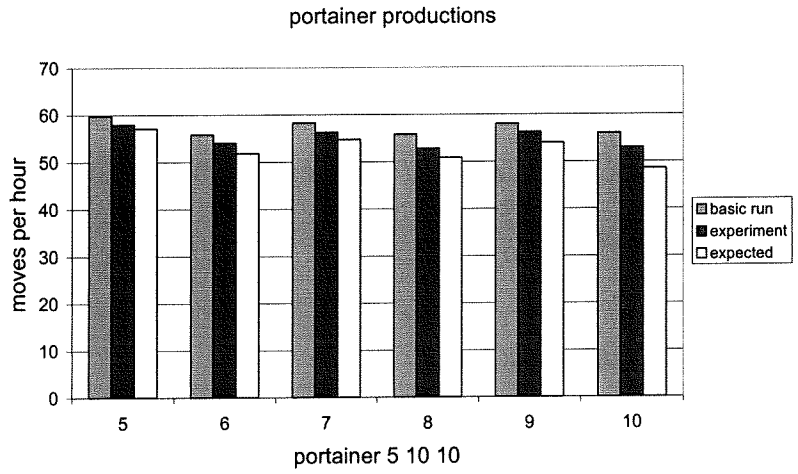


Figure 8.55, portainer productions

In the case of Berth 2 the big disturbances do not disturb the unloading process at the terminal too much. Obviously they only result in longer queues under the portainer locations. With a maximum row length of 6 while there are only 6 rgv's per portainer used during unloading. The queues under the portainers are shown in Appendix 25.

When a portainer is loading a vessel with containers from an adjacent berth, as is the case on Berth 1 & 3, a big disturbance could prove to be a bigger problem because more rgv's per portainer are used in this process. Figure 8.56 shows a result of the experiment. At one point a queue of 8 rgv's arises. This is the result of a big disturbance of 500 seconds. The queue length of 5 rgv's in the picture is the result of a disturbance of 200 seconds.

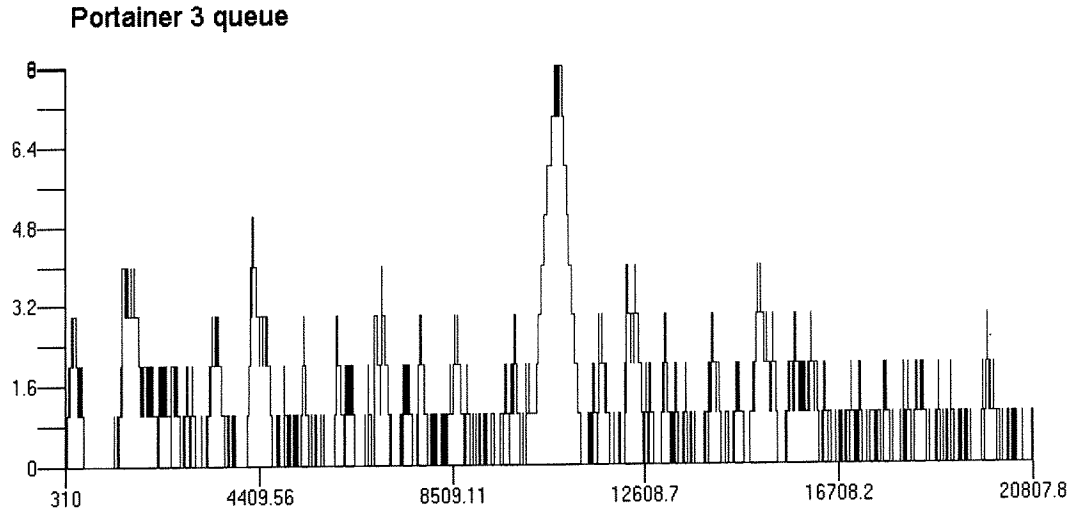


Figure 8.56, rgv queue under portainer 3

This second disturbance is shown in together with the rgv queue which is created by the disturbance.

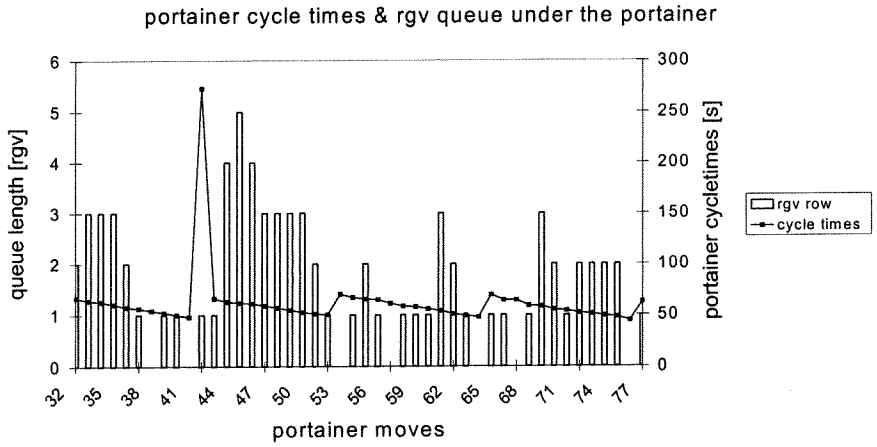


Figure 8.57, reaction to a big disturbance

The line shows the disturbance in the cycle times and the columns show the rgv waiting queues under portainer 3.

8.6.3 Stacking crane cycle times

The crane cycle times used at the reference terminal have deterministic cycle times of 80 seconds. In reality these cycle times will be different depending on the location of the containers which have to be retrieved or stacked. These cycle times can differ from 44 seconds for a container to be retrieved from the first floor up to 116 seconds for a container that has to be delivered to the top of a stack.

This is a reason to experiment with a triangular function for the crane cycle times. This will be a triangular function with a minimum of 44 [s], a maximum of 116 [s] and a mean of 80 seconds. The results of this run will be compared with the results of the reference terminal. Important performance parameters during this run are agv queues in the stacks and under the portainers and the portainer productivity.

Portainer productivity
The portainer occupancy rates in the second run are on the average 3.5% lower and the mean portainer cycle time is the same. This means the production will be 3.5% less. The portainer occupancy rates are shown in Figure 8.58, portainer occupancy rates of the experiment compared to the basic run.

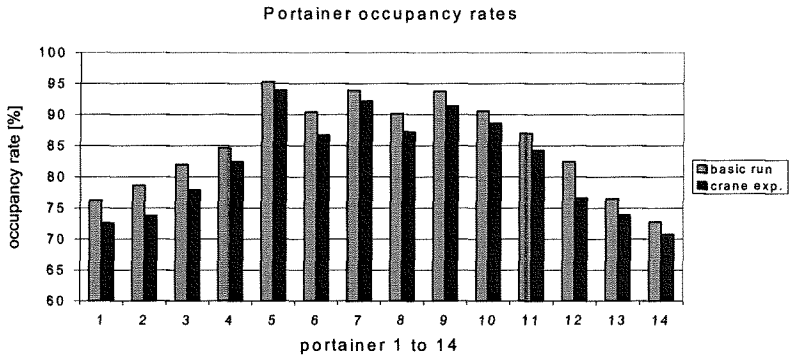


Figure 8.58, portainer occupancy rates of the experiment compared to the basic run

The productions for the portainers are shown in Table 8.12.

	Number of containers	Production per portainer (moves/hour) basic run	Production per portainer (moves/hour) crane exp.	Difference in handling time [hours]	Difference in handling time [%]
Berth 1	1050 loading	49.9	47.5	0.2	4.8
Berth 2	3150 unloading	57.3	55.7	0.2	2.8
Berth 3	1050 loading	49.4	47.3	0.2	4.3

Table 8.15, portainer productions during the basic run and the crane experiment

The difference is about 12 minutes at each berth.

Rgv waiting queues on the warehouse lanes
The warehouse lanes in the 20' warehouses in Yard 2 are the busiest of the terminal. These queues sometimes exceed the length of 3 rgv's. There is even a case where the queue length exceeds the length of 4 rgv's, which is not acceptable. This queue is shown in Figure 8.59.

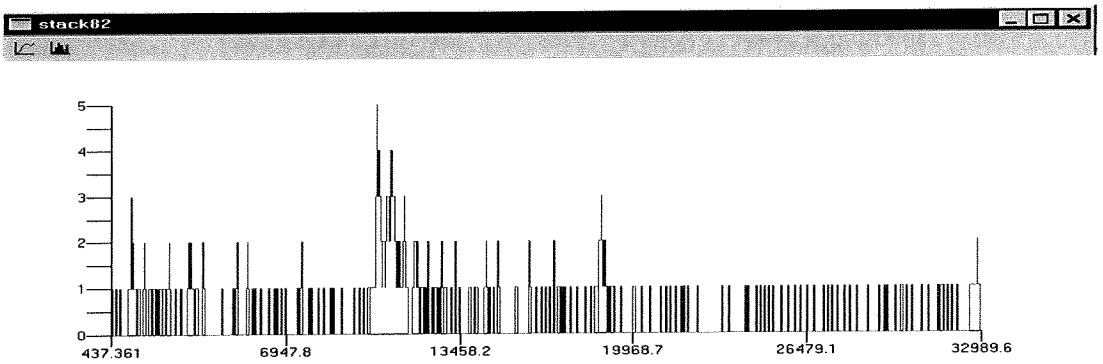


Figure 8.59, the rgv queue on warehouse lane 82

The queues in the 40' stacks never exceed the length of 3 rgv's. The queues on the warehouse lanes are shown in appendix 26.

9 Discussion of the Simulation Results

The results of the simulation runs will be discussed in this chapter. Adding some critical notes will help to see the optimistic results in some perspective.

9.1 Introduction

The results of this study are the outcome of strongly schematised simulation runs. The parts of the simulation runs that have been most strongly schematised are the portainer handling rates and the warehouse crane handling rates. Assumptions concerning the properties of the rgv's and the stacking height in the warehouses have to be regarded with caution. Each of these items will be described here.

Portainer handling rates
As has been illustrated in Chapter 4 the portainer handling rate varies with the location of the container in the hold of a vessel. The variable handling rates have been taken from FAMAS [L.12]. Apart from these variations the simulation runs concerning the reference terminal have not been carried out with other irregularities. Some of the most important irregularities that have been left out of the simulation are:

- Time loss due to jammed containers in the hold of the vessel.
- Time loss due to the removing of the hatches from the vessels.
- Time loss due to the lateral movements of the portainers.

Apart from these time losses the portainers have been modelled in such a way that each portainer will unload more or less the same amount of containers. This schematisation has a positive effect on the service time of a vessel since in reality the containers are not distributed over the holds in such a favourable way.

Warehouse crane handling rates
The average handling cycle of a warehouse crane has been determined in Appendix 8. For a 13-storey warehouse the average time it takes for a warehouse crane to complete a handling cycle amounts to 80 seconds. This average handling rate is the deterministic value of the handling rate of a warehouse crane in the simulation model. This schematisation excludes the variations in handling rates that will take place in reality. Effects that might occur due to these variations, like extra rgv congestion resulting in time loss, have consequently been left out of the analysis.

Stacking height
The stacking height in the warehouses has been based on an assumption that it is possible to build a 13-storey warehouse and a warehouse crane able to handle containers over these heights. A study has to be carried out if such a container warehouse can be realised. If a 13-storey warehouse can not be realised the consequence of this assumption is double. First, lower warehouses will result in more required land area, and second, the rgv delivery and retrieving cycles will demand longer transportation cycles. This means more rgv's have to be assigned per portainer if the vessel handling rates are to be kept constant.

Rgv properties

The properties of the rgv's have been assumed to a certain extend. The time required to complete the 90° rotation of the wheels of the rgv during turning has been assumed to be 4 seconds. Furthermore the acceleration of the rgv's has been assumed to reach 1 m/s^2. If these assumptions prove to be incorrect rgv cycles will require more time resulting in longer service times for the container vessels or more rgv's per portainer.

The results obtained from the simulation runs may thus be optimistic. In order to obtain more lifelike and realistic results the above-discussed issues should be taken into account. That makes the character of this study a more orientating and academic one. These results can not be expected to be matched in reality.

Nevertheless, the results are encouraging and will be discussed in the next parts of this chapter. The outcome of the simulation runs on the reference terminal must be regarded as a first simulation round. By adding disturbances to the handling cycles of the portainers and warehouse cranes, the influences of these disturbances can be compared to the simulation runs of the reference terminal used in this study.

9.2 Interpretation of the simulation results

In this paragraph the performance indicators of the simulation runs as dealt with in Chapter 8 will be interpreted and discussed.

9.2.1 Handling performance

The handling performances resulting in the vessel service times for both feeder vessels and mega vessels have been illustrated below.

The displayed results contain the handling rates and service times during the unloading and the adjacent loading procedure. The reduction on the service time of a container vessel gained by the shuffling procedure is not considerable compared to the time needed to complete the adjacent loading procedure. Especially if the number of adjacent loading rgv's per portainer is increased to 12 rgv's per portainer.

The shuffling procedure will therefore not be taken into account any further. Table 9.1 gives an overview of the simulation results.

	Feeder Vessel						Mega Vessel					
	Handling rates (mvs./hour)			Service times (hours)			Handling rates (mvs./hour)			Service times (hours)		
	Unl	Adj	Average	Unl	Adj	Total	Unl	Adj	Average	Unl	Adj	Total
Undisturbed	236	206	221	4.5	5.1	9.6	352	324	338	8.9	9.7	18.6
Disturbed	231	194	213	4.5	5.4	9.9	344	300	322	9.2	10.5	19.8

Table 9.1, the handling rates and service times

Undisturbed unloading procedure

Compared to the preliminary calculation made in Chapter 3.6 the handling rates of the undisturbed simulation runs during unloading do not differ greatly for both the feeder vessels and mega vessels. This can be seen in Table 9.2.

Unloading procedure	Feeder vessel		Mega vessel	
Preliminary calculation	240 mvs/h	100 %	360 mvs/h	100 %
Undisturbed simulation	236 mvs/h	98 %	352 mvs/h	98 %

Table 9.2, comparison between undisturbed situation and the ideal preliminary calculation

The main reason that these deviations are relatively so small is because the simulation runs have been carried out without any great disturbances in the portainer cycles or in the warehouse crane cycles. The undisturbed simulation runs can therefore be regarded as ideal runs. An important conclusion that can be drawn however is the fact that the horizontal transportation turns out not to be a bottleneck when the portainers and warehouse cranes are operating at full capacity. The amount of 6 rgv's operating for a portainer during the unloading procedure is satisfactory as well.

It is recommended to include handling cycle irregularities to the simulation program and compare the outcome to the results of the reference terminal that has been used in this study.

Undisturbed adjacent loading procedure

The simulation runs concerning the adjacent loading procedure show a greater deviation compared to the preliminary calculation as can be seen in Table 9.3.

Adjacent loading procedure	Feeder vessel		Mega vessel	
Preliminary calculation	240 mvs/h	100 %	360 mvs/h	100 %
Undisturbed simulation	206 mvs/h	86 %	324 mvs/h	90 %

Table 9.3, comparison between undisturbed situation and the ideal preliminary calculation

The handling performances on the terminal during adjacent loading are relatively somewhat lower than during the unloading procedure. As was the case during the unloading procedure great disturbances have not been included to the handling rates of the portainers and the warehouse cranes. The main reason that the handling rates are relatively lower than during the unloading procedure is because the occupancy rates of the portainers are lower than during unloading caused by operating with 10 rgv's per portainer during the adjacent loading procedure. This is a result. The adjacent loading simulation experiment in Paragraph 8.6.1 with 12 rgv's per portainer confirms this conclusion as the handling rates increased.

A recommendation is simulating the adjacent loading procedure with more than 10, preferably 12 rgv's, and including random disturbances in both portainer and warehouse crane handling cycles.

Disturbed unloading procedure

By carrying out simulation runs handling with than one container vessel the occupancy of the terminal has been increased. These runs are called the disturbed runs as opposed to the undisturbed runs which are carried out with only one moored vessel.
As is the case during the undisturbed unloading procedure the handling rates are still relatively high. This can be seen in Table 9.4.

Unloading procedure	Feeder vessel		Mega vessel	
Preliminary calculation	240 mvs/h	100 %	360 mvs/h	100 %
Disturbed simulation	231 mvs/h	96 %	344 mvs/h	96 %

Table 9.4, comparison between disturbed situation and the ideal preliminary calculation

As is the case with the undisturbed situations the handling cycles of the portainers and the warehouse cranes have not been penalised with random disturbances. The reason for the decrease of the handling production therefore is the increased numbers of rgv's operating on the different berths. More operating rgv's means a higher chance for a rgv to wait on lane crossings.

It is recommended to include handling cycle irregularities to the simulation program and compare the outcome to the results of the reference terminal that has been used in this study.

Disturbed adjacent loading procedure

The simulation runs concerning the adjacent loading procedure show a greater deviation compared to the preliminary calculation as can be seen in Table 9.5.

Adjacent loading procedure	Feeder vessel		Mega vessel	
Preliminary calculation	240 mvs/h	100 %	360 mvs/h	100 %
Disturbed simulation	194 mvs/h	81 %	300 mvs/h	83 %

Table 9.5, comparison between disturbed situation and the ideal preliminary calculation

Comparison between the results of the simulation runs of the undisturbed and the disturbed variants of adjacent loading shows a decrease of production of a few percentages. The reason for this decrease is the increase of operational rgv's on the terminal. Increasing the number of rgv's per portainer will further increase the occupancy rates of the portainers and thus the handling capacity per berth.

A recommendation is to simulate the adjacent loading procedure with more than 10, preferably 12 rgv's, and to include random disturbances in both portainer and warehouse crane handling cycles. These recommendations have been carried out in separate simulation runs and will be discussed in the next paragraph.

9.2.2 Influence of variations to the reference terminal

Three experiments have been conducted in Paragraph 8.6 concerning variations to the reference terminal. One experiment concerning the adjacent loading with 12 rgv's per portainer, one with variable crane handling cycles and one with the addition of random disturbances to the portainer handling cycles. All these experiment have been conducted to the normative situation on the terminal. Integrating all three these experiments into 1 simulation run will show the amplifying effects they have on each other and is therefore recommended.

Simulation with 12 rgv's during adjacent loading

What stands out in this discussion is the fact that the horizontal transportation operating on the terminal has a high capacity, especially during unloading. If the amount of operational rgv's per portainer is increased to 12 during an adjacent loading procedure the portainers will achieve occupancy rates comparable to the occupancy rates achieved during the unloading procedure. This means that the horizontal transportation units, the rgv's, are able to cope with the high production rates of the portainers. Simulation experiment I in Chapter 8.6 illustrates this.

Simulation with additional disturbances to the portainer handling cycles

The added disturbances to the portainer handling cycles have an influence on the production capacity of the terminal. The higher the mean handling cycles will be due to these disturbances the lower the mean handling production will be.

An interesting result however is the recovery rate of the rgv waiting queue under the portainers. Logically, the rgv waiting queue will increase as a portainer handling cycle is penalised with a long disturbance. The recovery rate is rather high as the rgv waiting queue slinks to normal proportions after a limited period of time. This can be seen in Figure 8.55 and 8. 56. This result shows that the horizontal transportation can be regarded as flexible.

Simulation with variable warehouse crane handling cycle times

Introducing a triangular warehouse crane cycle function has a negative effect on the container vessel service time, because these irregularities create additional waiting times for the rgv's on the warehouse lanes. The waiting queues will grow longer if for instance a warehouse crane receives a number of longer handling assignments following each other, as can be seen in Figure 8.58 in Chapter 8.

The time loss caused by this experiment has been quantified and shown in Table 9.6.

		Basic reference terminal simulation run			Variable crane handling cycle simulation run			Difference in handling time	
	Number of containers	Production per portainer (mvs/hour)	Production per berth (mvs/hour)	Handling time (hours)	Production per portainer (mvs/hour)	Production per berth (mvs/hour)	Handling time (hours)	ΔT hours	ΔT %
Berth 1	1050 loading	49.9	199.6	5.3	47.5	190	5.5	0.2	4.8
Berth 2	3150 unloading	57.3	343.8	9.2	55.7	334	9.4	0.2	2.8
Berth 3	1050 loading	49.4	197.6	5.3	47.3	189	5.5	0.2	4.3

Table 9.6, portainer productions of the basic run and the crane experiment

The introduction of variable handling cycles for the warehouse cranes seems not to influence the service times to a dramatic extend. The adjacent loading time will grow with about 4.5 % on Berth 1 & 3, while the unloading procedure on Berth 2 will grow to about 2.8 % on Berth 2. It must be noted however that although the simulated situation on the terminal is a heavy occupied one, more runs should be carried out in order to obtain a proper understanding on the influence of this variation to the reference terminal.

9.2.3 Crane occupancy rates

The normative situation from which the occupancy rates of the warehouse cranes are the highest is the adjacent loading on all 3 berths as is stated in Paragraph 8.5.5. In Table 9.7 these crane occupancy rates together with the number of cranes that are active on the reference terminal have been plotted.

	Yard 1		Yard 2		Yard 3	
	20' stack	40' stack	20' stack	40' stack	20' stack	40' stack
Occupancy rate (%)	43.3	29.0	34.7	23.2	43.3	29.0
Number of warehouse cranes	8	12	12	18	8	12

Table 9.7, number of cranes and their occupancy rates

As the warehouse cranes are able to transfer trough the warehouse rows, they can be concentrated to locations where the demand for production is highest. The transfer possibility increases the flexibility of the container storage system and offers a possibility to make more efficient use of the warehouse cranes. The occupancy rates of the warehouse cranes on the reference terminal are rather low as can be seen in Table 9.7.

Assuming that a warehouse crane occupancy rate of 70 % is acceptable allows determining the number of needed warehouse cranes. This has been calculated in Paragraph 8.5.5. The amount of warehouse cranes operating in 20' container stacks is 16. As the 20' container stacks consist of 2 warehouse rows, each of these rows will host 8 warehouse cranes.

The 40' container stacks will host 18 warehouse cranes. This amount of cranes will be spread out over 3 warehouse rows resulting in 6 warehouse cranes per row.

It is recommended to carry out simulation runs with the established amount of cranes.

9.2.4 Consequences for the terminal area

The amount of land area that the reference terminal occupies is based on a terminal layout in which the rgv waiting rows on the crossings have not been included. By carrying out an analysis on the waiting row length for a number of nodes the required space can be determined.

The required land area for the reference terminal is 178 * (1000-132) ≈ 15.5 ha as can be seen in Appendix 1. The required rgv parking space behind the yards is not included.

9.2.5 Integration of rgv waiting rows on the terminal

The land usage of the final terminal in this study is the land usage needed for the simulated reference terminal plus the addition of space needed for the rgv's waiting rows on the nodes. From the rgv waiting row lengths, included in the appendices, an impression can be obtained for the needed space per node. The normative situation of the reference terminal for the node rows is the adjacent loading on all 3 berths. This is Experiment 1 dealt with in Paragraph 8.5.1.

The maximum required rgv waiting row length between 2 inter-berth lanes, has been plotted in Table 9.8.

	Location of the waiting rgv row	Maximum waiting row length found during the simulation run (number of rgv's)	Applied waiting row length on the terminal (number of rgv's)
Inter-berth lanes located at the top of the terminal	Between inter-berth lanes 1 & 2	3	6
	Between inter-berth lanes 2 & 3	3	6
	Between inter-berth lanes 3 & 4	5	9
	Between inter-berth lanes 4 & yard	5	9
Inter-berth lanes located at the bottom of the terminal	Between inter-berth lanes yard & 5	3	9
	Between inter-berth lanes 5 & 6	4	9
	Between inter-berth lanes 6 & 7	4	6
	Between inter-berth lanes 7 & 8	4	6

Table 9.8, simulation results for the waiting rows and applied waiting row length

The maximum waiting row length found during the normative simulation run does not occur very often as can be seen in Appendix 20. However, the fact that longer rgv waiting rows have not been found during this simulation run does not mean that the rows will never exceed the found lengths.

Because the rgv waiting row lengths have not been statistically analysed the applied rgv waiting row lengths have been chosen rather arbitrary. The chance that the rgv waiting rows on the reference terminal will exceed the applied permitted length is low. Introducing random disturbed operating times for the portainers and the warehouse cranes though, will probably lead to longer rgv waiting rows. It is therefore recommended that a simulation run should be

carried out during the adjacent loading on all 3 berths with random disturbed operating times for portainers and warehouse cranes.

If the rgv waiting row length exceeds the applied permitted rgv waiting row length nonetheless, the rgv will have to drive past the crossing point and request an alternate route. Another consideration to be made is the fact that the destinations of the rgv's are completely randomly chosen. By introducing a more structured rgv destination choosing system potential high-density situations for rgv's can be prevented.

The integration of the rgv waiting row space into the reference terminal has been shown in Figure 9.1. The available space between inter-berth lanes 1 and 2 and inter-berth lanes 2 and 3 is 6 rgv's, while the available space between inter-berth lanes 3 and 4 and inter-berth lane 4 and the yard is 9 rgv's. The same space has been reserved for the bottom inter-berth lanes. The complete top view of the terminal can be seen in Appendix 29.

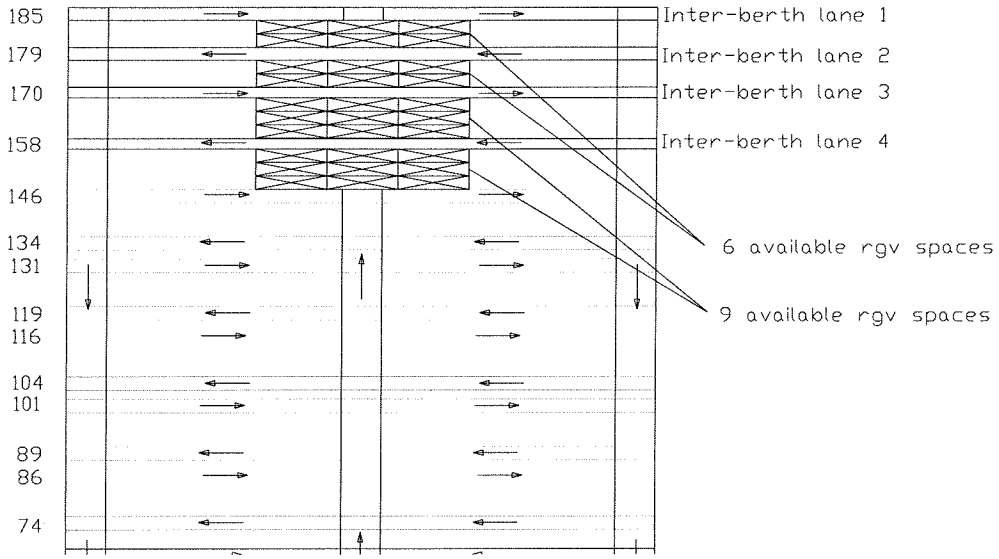


Figure 9.1, available waiting spaces between the inter-berth lanes

By integrating the waiting spaces for the rgv's into the reference terminal, the rgv travelling cycles will increase. This will probably not affect the results to much as the increase of travelling distance is relatively small.

The total of needed land area has increased to 185 * 884 ≈ 16.5 ha. This is the area without rgv parking and service buildings.

9.3 Queuing theory

In order to determine the annual berth production of the terminal longer runs are required. This model is to detailed to simulate these longer periods. Using results of the reference terminal runs linked to vessel arrival schemes could say something over the turn around times of the vessels and the vessel queues in the anchorage over a longer period of time.
The queuing system can be described by the inter arrival distribution of the vessels, the distribution of the service times and the number of berths. As this study concerns a none existing terminal some assumptions concerning the arrival times of the vessels must be made.
For the mega vessel berth and the two feeder berths two situations have been examined, using different arrival distribution functions.

9.3.1 Mega vessel berth

Concerning the mega vessel two situations using different arrival time and service time distributions are looked at.

M/D/1
- Negative exponential inter arrival time distribution
- Deterministic service time distribution
- 1 Berth

$E_k/E_l/1$
- Erlang distribution for the inter arrival times
- Erlang distribution for the service times
- 1 Berth

M/D/1
The negative inter arrival distribution (N.E.D.) is used to model inter arrival times when arrivals are completely random.
The service times are assumed to have no variability, that is σ^2 =0, which means all the service times are assumed the constant value $1/\mu$.

If assumed that the service time of a mega vessel including the mooring is 22 hours and on the average 1 vessel per 36 hours arrives at the terminal, the occupancy rate, average waiting time and average waiting queue can be calculated.

$$\rho = \frac{\lambda}{\mu} \qquad (9.1)$$

where:

μ=service rate
λ=arrival rate

The occupancy rate in this case is (1/36)/(1/22) = 61%;
The average number of customers in the waiting queue N_w can be calculated using 9.2.

$$N_w = \frac{\rho^2}{2(1-\rho)} \qquad (9.2)$$

The average number of mega vessels in the queue is 0.48.
The average waiting time for these vessels W, can be calculated using formula 9.3;

$$W = \frac{\lambda \mu^{-2}}{2(1-\rho)} \qquad (9.3)$$

The average waiting time per mega vessel 17.23 hours.
The average turn around time for a mega vessel in this system using these values will be 39.23 hours.

$E_k/E_l/1$

In reality the arrival pattern of the mega vessel will not be completely random. Neither will the service times be completely deterministic.

The distribution of the arrival times of the vessels at the terminal is assumed to have an average of 36 hours and a standard deviation of 12 hours. The service times of the vessels have an average of 22 hours and a standard deviation of 6 hours. the distributions can be assumed to have an Erlang distribution.

While the probability equations are not algebraically soluble the values of;

- utilisation (u)
- variability of the E_k distribution of the inter arrival times (ν_a=(s.d/mean)^2)
- variability of the E_l distribution of the service times (ν_s=(s.d/mean)^2)

can be obtained from tables.

The utilisation of the berth is 61%.

The coefficient of variation of the arrival time;

$$\frac{\text{s.d}}{\text{mean}} = \frac{12}{36} = 0.33 \qquad (9.4)$$

so ν_a = (c.v)^2 of arrival time = $(0.33)^2$= 0.109.

The coefficient of variation of service time;

$$\frac{6}{22} = 0.27 \qquad (9.5)$$

so ν_s= (c.v)^2 of service time = $(0.27)^2$= 0.074

From the tables of waiting times (table III in Appendix 27) the average waiting time in units of average service time (w) can be read. Linear interpolation on ν_a gives w = 0.0996. The average waiting time per mega vessel would be 22*0.0996= 2.19 hours.

The occupancy rate of the Mega vessel berth during these two situations is 61% and rather high. Figure 9.2 shows the average waiting time of a vessel, in units of average service time, in the queue in relation to the occupancy rate of the berth.

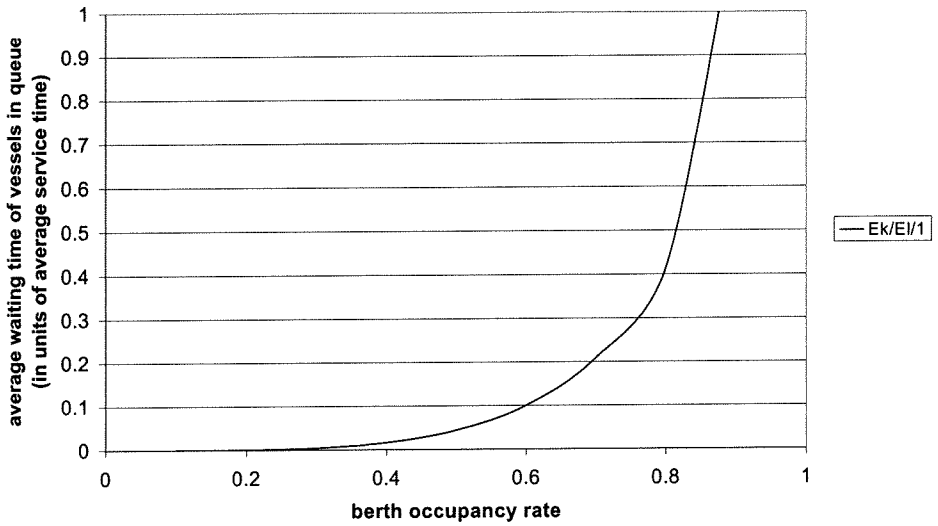


Figure 9.2, The relationship between average ship waiting time and the berth occupancy rate

9.3.2 Feeder berths

Concerning the feeder berths a few situations using different arrival time and service time distributions are looked at and approximations of waiting times are given.

M/D/2
- Negative exponential inter arrival time distribution
- Deterministic service time distribution
- 2 Berths

$E_k/E_l/2$
- Erlang distribution for the inter arrival times
- Erlang distribution for the service times
- 2 Berths (n=2)

M/D/2
The negative inter arrival distribution (N.E.D.) is used to model inter arrival times when arrivals are completely random.
The service times are assumed to have no variability, that is σ^2 =0, which means all the service times are assumed the constant value $1/\mu$.
It is assumed that the service time of the feeder is 12 hours including the mooring, and on the average 1 feeder arrives in 12 hours. Values of waiting times can be obtained from tables.

The utilisation is;

$$u = \frac{\rho}{n} = 50\% \qquad (9.6)$$

From table I can be obtained that the w=0.176. The average waiting time will be w * average service time = 2.12 hours.

The average turn around time of a feeder will be 14.12 hours.

Ek/El/2
In reality the service times of the feeders will not be deterministic. Especially with variation in the number of containers per feeder the service times will differ. The average service time is assumed to be 12 hours, including the mooring with a standard deviation of 5 hours.
In this scenario the average inter arrival time of the feeders is 12 hours with a standard deviation of 7 hours.

An approximation of the value of average waiting time is given based on the linear interpolation on ν_a and ν_s using the queuing systems: M/M/2, D/M/2 and M/D/2. Let $W_2(\nu_a,\nu_s,u)$ be the average waiting time in $E_k/E_l/2$ with utilisation μ, $k^{-1} = \nu_a$ and $l^{-1}=\nu_s$, then

$W_2(1,1,u)$ = the average waiting time in M/M/2 with utilisation u;
$W_2(0,1,u)$ = the average waiting time in D/M/2 with utilisation u;
$W_2(1,0,u)$ = the average waiting time in M/D/2 with utilisation u;

The average waiting time in general case is given by:

$$W_2(\nu_a,\nu_s,u) = (1-\nu_a)*\nu_s*W_2(0,1,u) + \nu_a*(1-\nu_s)*W_2(1,0,u) + \nu_s*\nu_a*W_2(1,1,u) \quad (9.7)$$

The average waiting time in M/M/2 can be calculated using formula 9.8.

$$W = \frac{1}{n*\mu} * \frac{\rho^n}{n!} * \frac{P(0)}{(1-\rho/n)^2} \quad (9.8)$$

where

$$P(0) = \left[1 + \rho + \frac{\rho^2}{2!} + \frac{\rho^3}{3!} \ldots + \frac{\rho^n}{n!(1-\rho/n)}\right]^{-1} \quad (9.9)$$

Using (9.8) and (9.9) the average waiting time $W_n(1,1,u)$ = 0.333 * average service time.

The average waiting time in D/M/2 can be obtained from table II, $W_n(0,1,u)$=0.065 * average service time.

The average service time in M/D/2 obtained from table I = 0.176 * average service time.

With:
- $\nu_a=(7/12)^2=0.34$
- $\nu_s=(5/12)^2=0.17$

in formula (9.7) results in;

$$W_2(\nu_a,\nu_s,u) = (1-0.34)*0.17*0.0649 + 0.34*(1-0.17)*0.176 + 0.34*0.17*0.333 = 0.076$$

0.076 * average service time =0.91 hours average waiting time per feeder.

9.3.3 Yearly throughput

Using the results of the queuing system, something can be said very roughly about a yearly throughput at the terminal. The Assumptions made are shown below;

Assumptions;
- The mooring time of the vessels is included in the service time
- Erlang distribution for the inter arrival times mega vessels (48;12)
- Erlang distribution for the service times of the of the mega vessels (22;6)
- Erlang distribution for the inter arrival times feeders (12;7)

- Erlang distribution for the service times of the of the mega vessels (12;5)
- 1 Berth available for the mega vessels
- 2 berths available for the feeder vessels
- production per portainer (moves per hour) 55.7 unloading and 50 loading
- Average number of moves per mega vessel (6300 loading, 3150 ;unloading, 3150, loading)
- 6 portainers per mega vessel
- 360 days per year 24 hours per day

Demand;
The service level = average waiting time/average service time: 0.1

Result;
Using these assumptions with the queuing theory the yearly throughput can be calculated;

360 (days)*24 (hours)= 8640 hours per year is the terminal operational. On the average once in the 36 hours a mega vessel arrives at the terminal. In total 240 mega vessels arrive during the year.
During each mega vessel call 6300 container moves are made. In total this is 6300*1.5=9450 mega TEU per mega vessel call.
In a year 240*9450=2,268,000 TEU is transhipped using the mega vessel berth.
On the other two berths the same amount is transhipped, the total amount would be 4,536,000 TEUs for the terminal or an average of 1,512,000 TEUs per berth.

The mean occupancy rate for the mega vessel berth is 61%, the utilisation of the feeder berths is 50%.
The service level for the mega vessels is:

$$\frac{2.19}{22} = 0.099$$

The service level for the feeder is:

$$\frac{0.91}{12} = 0.075$$

These service levels are within the boundary condition.

Compared to the most advanced existing transhipment terminal these values are high. For example:
Every berth at the Pasir Panjang Terminal in Singapore handled an average of 750,000 TEUs in the year 1998.

10 Conclusions

The reference container terminal as proposed in this study can be realised from a logistic point of view.

- The horizontal transportation system has not shown unacceptable congestion rates during any of the simulation runs based on the reference terminal.
- The container storage system has shown no unacceptable congestions of rgv's waiting on the warehouse lanes.
- The length of rgv row's under the portainers is acceptable while maintaining a high occupancy rate for the portainers.

Conclusions for each component on the terminal are as follows:

Handling strategies
The optimal handling strategy for the reference terminal is unloading a container vessel in the yard located behind its berth and loading a container vessel from the yard located behind the adjacent berth.

The shuffling of containers is a time consuming process while the advantages are small. The difference between the adjacent loading and the fast loading of a container vessel are marginal.

Berth productions on Berth 1 & 3
The undisturbed berth productions on the reference terminal for the feeder vessel berths vary from 236 moves per hour for unloading to 206 moves/hour for loading from the adjacent berth, Yard 2. This production rate results in a service time of 9.6 hours for the standard feeder vessel used in this study.

The berth productions on the reference terminal for the feeder vessel berths during maximum occupation on the terminal vary from 231 moves/hour for unloading to 194 moves/hour for loading from the adjacent berth, Yard 2. This results in a maximum service time of 9.9 hours for the standard feeder vessel.

The handling rates on the feeder vessel berths have been achieved with 4 portainers per berth.

Berth productions on Berth 2
The undisturbed berth productions on the reference terminal for the mega vessel berth vary from 352 moves per hour for unloading to 324 moves/hour for loading from the adjacent yards, Yards 1 or 3. This production rate results in a service time of 18.6 hours for the standard mega vessel used in this study.

The berth productions on the reference terminal for the mega vessel berths during maximum occupation on the terminal vary from 344 moves/hour for unloading to 300 moves/hour for loading from the adjacent berth, Yards 1 or 3. This results in a maximum service time of 19.8 hours for the standard mega vessel.

The handling rates on the mega vessel berth have been achieved with 6 portainers.

Yearly throughput
Compared to the most advanced existing transhipment terminal the calculated yearly throughput of 1,512,000 TEU per berth is very high. For example:
Every berth at the Pasir Panjang Terminal in Singapore handled an average of 750,000 TEUs in the year 1998.

Number of rgv's
6 rgv's per portainer during the unloading of a container vessel is sufficient, as the portainer occupancy rates remain high. During the loading of a container vessel from an adjacent yard the number of 10 rgv's is low. 12 rgv's per portainer will provide higher portainer occupancy rates thereby increasing the handling rates.

Warehouse cranes
The reference terminal is equipped with 70 warehouse cranes. An occupancy rate calculation has been made based on the output of the computer simulations. By applying a 70 % occupancy rate to the warehouse cranes during the normative configuration, this results to 16 warehouse cranes in the 20' stacks and 18 warehouse cranes in the 40' stacks.

Applying variable handling cycle times with a mean cycle of 80 seconds instead of a deterministic handling cycle of 80 seconds decreases the terminal productions with about 4 %.

Land usage
The land usage of the reference terminal as simulated and without space reserved for the rgv waiting rows is 175.5 m * 884 m = 15.5 ha.

The land usage of the terminal included the rgv waiting capacities is 185 m * 884 m $\approx$ 16.5 ha.

This is the area without rgv parking and service buildings.

11 Recommendations

Annual berth production
In order to determine the annual berth production of the terminal longer simulation runs should be initiated. This model is too detailed to simulate these longer periods. The level of detail should be reduced before periods as long as months can be simulated.

- The actual driving of the rgv's on their lanes should not be included in the model as this asks too much of the memory space of the computer simulation program. The driving could be replaced by time penalties. The duration of these time penalties depends on the situation on the terminal at that moment. Their values can be gathered from runs of the current model.
- The terminal must be linked to a vessel arrival scheme to determine the turn-around times of the vessels and to determine the needed stack capacities.
- A simulation run with less than the maximum number of 14 portainers can be carried out carrying out the portainer assigning process described in Chapter 6.2.5.

PROSIM should be able to be supported by animation, which would make it more apprehensible for outsiders. The effects of different parameter changes are than easily spotted. For the current model animation is not an option as the program runs too slow.

Based on these longer runs an analysis can be made to determine the needed number of portainers and the needed stack capacity.

Cost calculation of the project
This study has not been concentrated on the costs. Obviously the costs of the project are very important. A cost analysis for this project including the civil structures needed for the terminal components like the foundation should be made in order to determine the feasibility of the terminal.

Properties of the rgv
In this study the acceleration of the rgv's is assumed to be 1 m/s^2. This is an assumption and research should prove whether this is feasible when using linear motor based transfer technology.

The disc rotation used to turn the wheels of the rgv when it has to make a 90^0 turn takes 4 seconds. This is an assumption, research should conclude whether this is possible.

Rgv-parking
Not all available rgv's will be active constantly. For this reason rgv parking is needed. The positioning should be determined and added to the layout of the terminal.

Horizontal transportation layout
The reference terminal layout consists of 2 pairs of inter-berth lanes. A simulation run with 1 pair of inter-berth lanes should determine if 2 pairs are necessary.

Container Warehouses
The maximum stacking height has, quite arbitrary, been assumed to be 12 containers high. A calculation for finding the economic optimum between stacking height and number of warehouses should be made. An important parameter is to what degree the land area is

scarce. If, as in Singapore, massive land reclaiming projects are needed, increasing stacking height will be cost effective.

Service points
The reference terminal contains 3 service points per stack-lane. These service points all have to host a shuttle in order to retrieve the containers from the visiting rgv's. A study should be made in order to find the optimum number of service points per warehouse lane.

Shuttles
In the terminal shuttles are used to transport a container between the warehouse crane, a service point and the rgv. The shuttle is assumed to transport a container in 15 seconds from a service point onto a warehouse crane, research should conclude whether this is possible.

Warehouse cranes
The reference terminal consists of 70 warehouse cranes. The occupation rate of these cranes is low. Calculations suggest the terminal could function well with less than 70 cranes if the cranes interchange more between the warehouses. An assumption has been made that an occupancy rate of 70 % is acceptable. Simulation runs should be carried out to find the maximum acceptable warehouse crane occupancy rate.

Portainer cycle times
This study has been carried out using portainer cycle times based on the FAMAS study. Different portainer cycle times of different portainer systems could be implemented in the program to check the reactions of the terminal.

Optimistic results
The simulation results are very optimistic as a result of only small disturbances in the terminal process. The process runs very smooth.

The warehouse crane cycle times in the experiments used are deterministic cycle times. The experiment with variable crane cycle times proved that a variable crane cycle time with the same mean does not produce the same handling rates on the terminal.
Different warehouse crane cycle times in combination with disturbances in the portainer cycle times and disturbances in the horizontal transport should be implemented in the program.

Twin lift
If a twin lift system is applied for the 20 feet containers the portainer production [TEU/move] could improve from 1.5 TEU/move to 2 TEU/move. The number of rgv's that transports 20' containers between the stacks and the berths will decrease. The warehouse crane occupancy rates will increase as a result.

Logistic control
As the terminal is almost completely automated, the control of the full range of terminal activities is a key condition for the realisation of this terminal. This is a very complicated logistic issue, which should be investigated for feasibility.

12 References

[L.1] Kremers, L.G.L., *Analyse van het proces van een containerkraan*, Delft 1996.

[L.2] The Motor ship by Kathy Oxtoby, *GL examines 8000 TEU strenghts*, December 1997.

[L.3] Elbers, A.W.& Aalten, E.J.P., *Behandelconcepten voor grote containerschepen*. Vakgroep Transporttechnologie, report 96.3.LT.4728, TU Delft, August 1997

[L.4] Drewry, *Post-panamax containerships 6000 TEU and beyond*, september 1996

[L.5] Carlebur, A.F.C.; Het FAMAS-programma, Schip & werf de zee, September 1998

[L.6] Ligteringen H.; Container handling in Seaports; Pianc conference, September 1997

[L.7] Bello A.E., Laffitte J.; Purchase, Operation and Maintenance of a Container Crane Spreader, Port Technology International.

[L.8] Jordan, M.A.; Super productive cranes, Lifetech consultants; TOC'97, Barcelona, June 1997.

[L.9] Groenveld, R.ir.; Service Systems in Ports and Inland Waterways f12/f13, Vakgroep Waterbouw, TU Delft September 1996.

[L.10] Nooijen F.; Automated guided vehicle system for ECT's Delta/Sea-Land Terminal, Terminal operations Conference, June 1992

[L.11] FAMAS-AGV cluster, FAMAS-AGV rapport fase 1, Benschop, 1998.

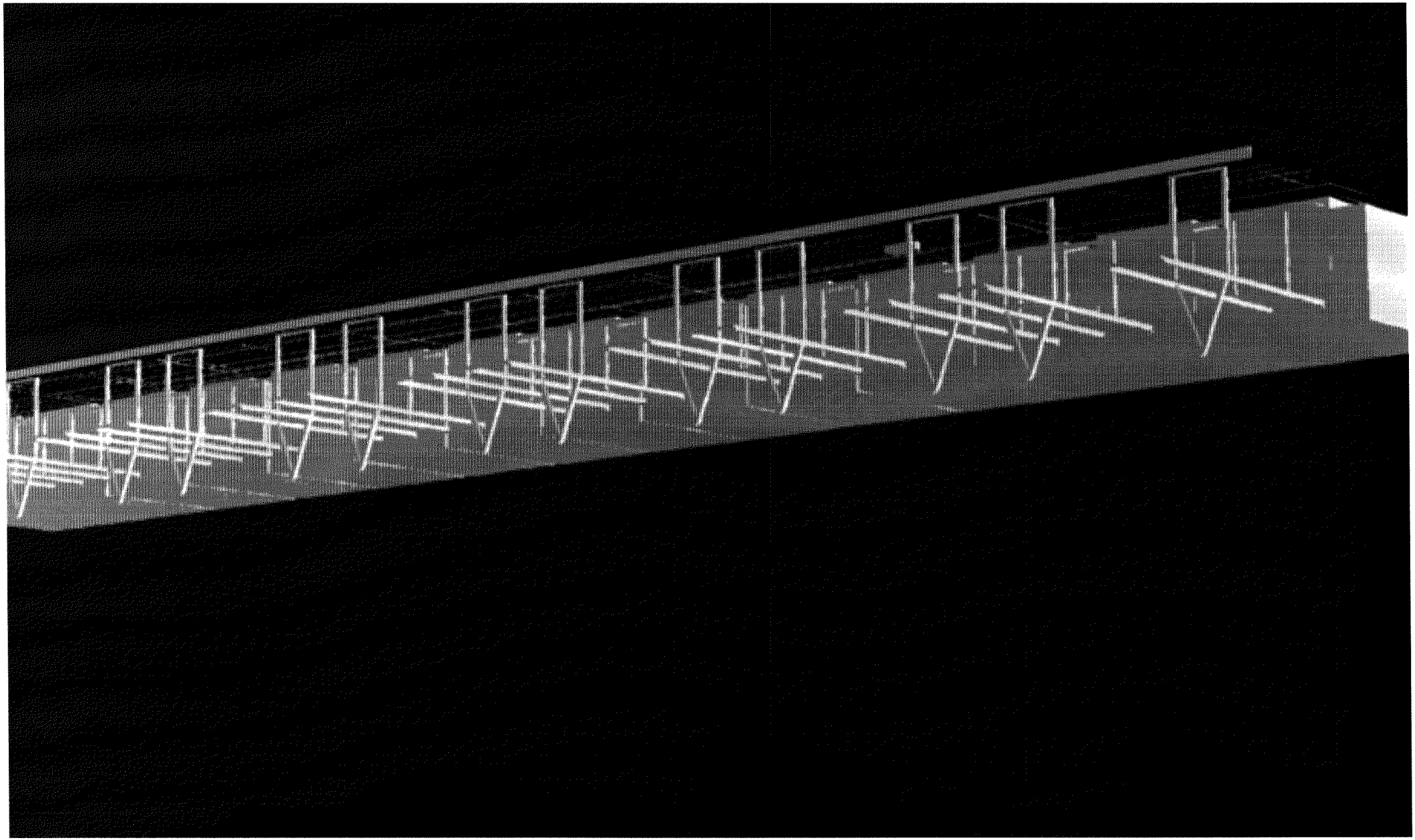
[L.12] Bakelaar, A.D.; Crane feeding system; Vakgroep Transporttechnologie, report 98.3.TT.5044; TU Delft June 1998

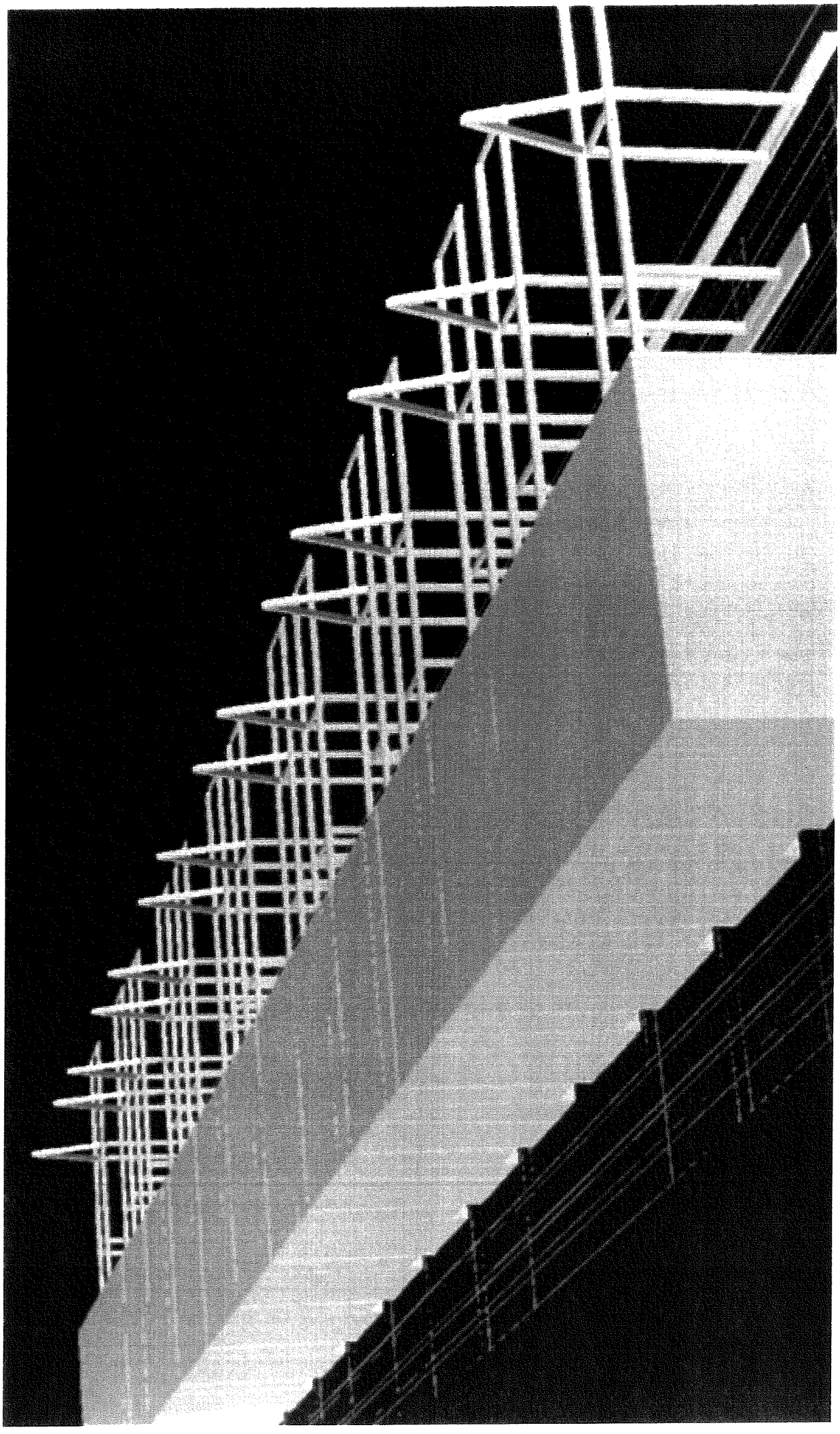
[L.13] Franke, K.P.; Container terminal visions become reality, Ports and waterways, September 1998.

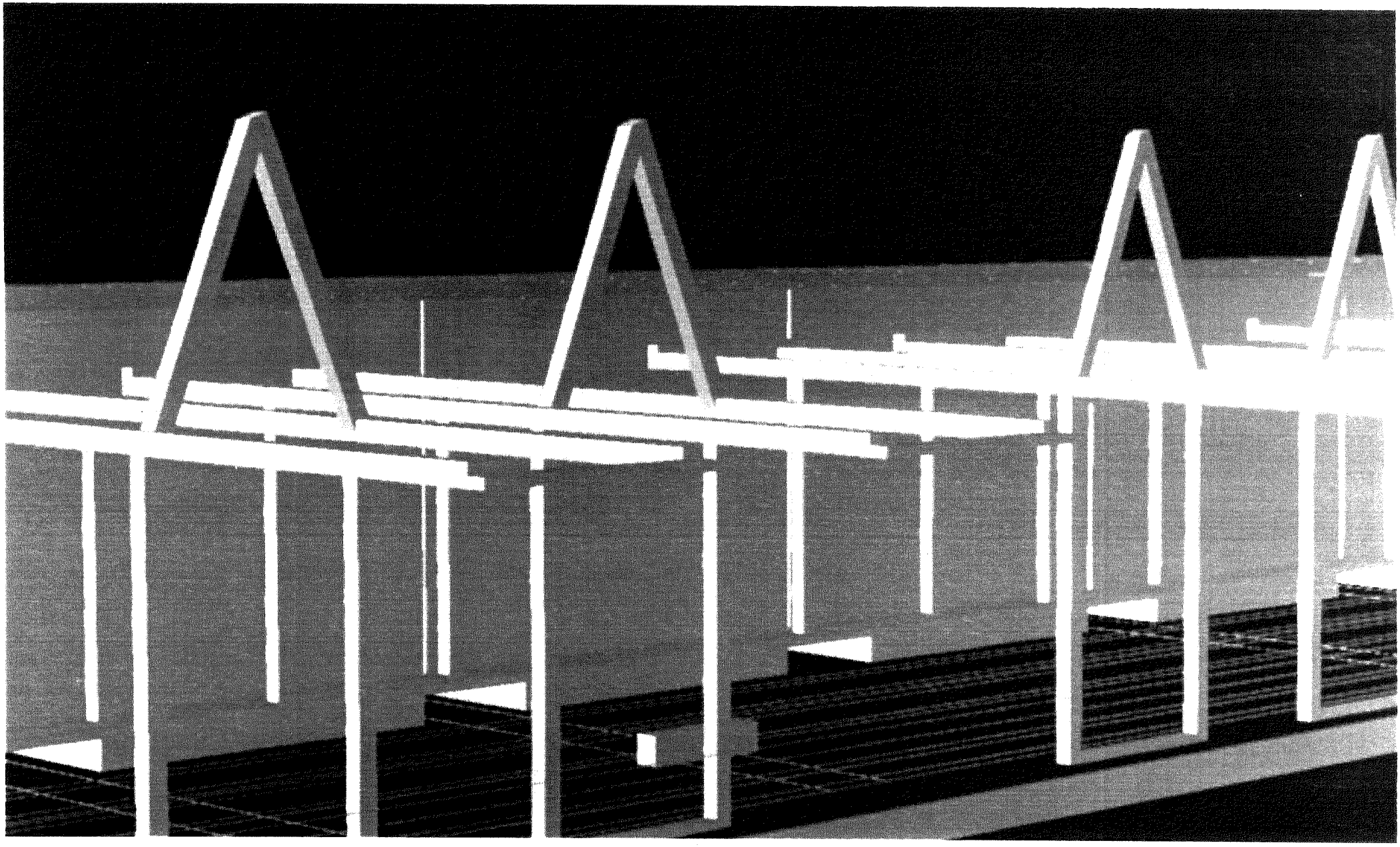
[L.14] Carder,F.B., a.o; Apparatus for marine cargo container handling and storage, U.S.patent number: 5.540.532, July 1996.

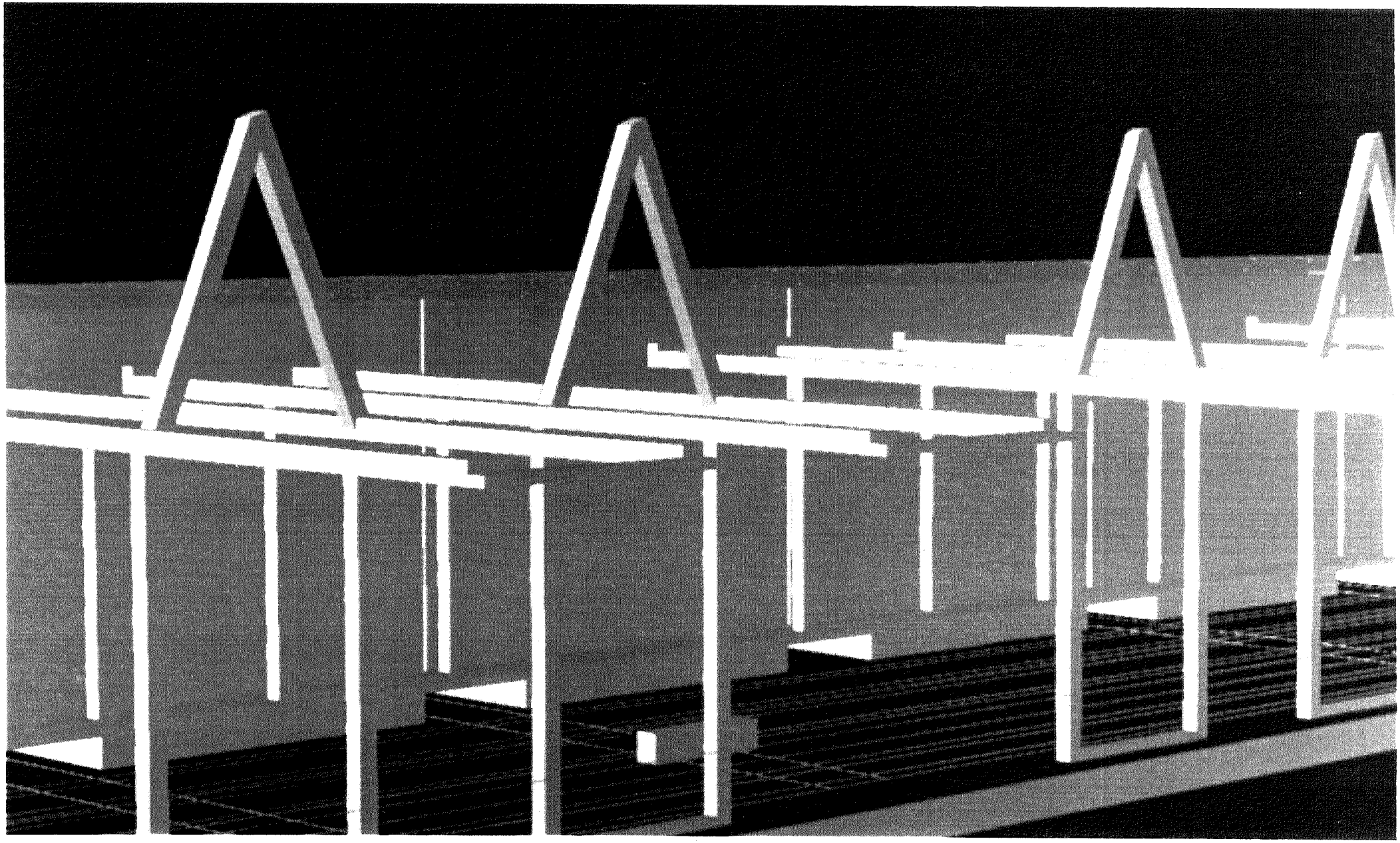
Internet site:
http://www.pacecocorp.com/page12.html

http://www.psa.com.sg



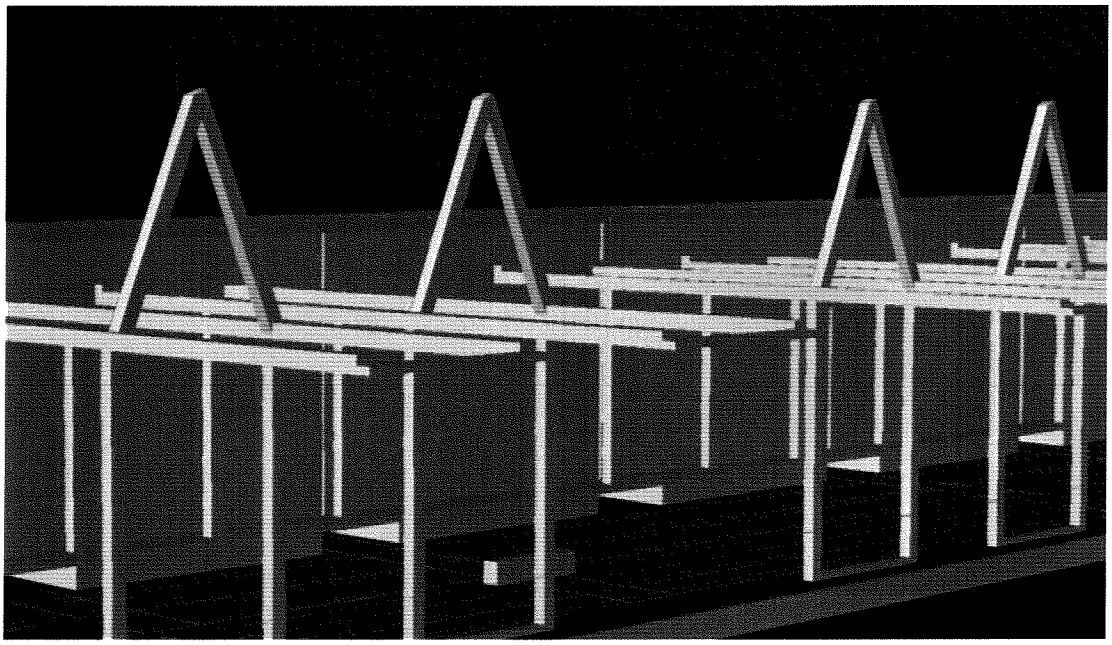






Study on the Design and Simulation of a High capacity transshipment terminal

- ***Appendices 1-27 & 29***



Impression of the portainers and container warehouse

December 2000
Hein van Hees
Thies Visscher

TUDelft
Delft University of Technology

Faculty of Civil Engineering and Geosciences
Department of Hydraulic and Geotechnical Engineering
Section of Hydraulic Engineering

Appendices

Appendix 1: Lay out of the reference terminal 3
Appendix 2: Node numbers and stacknumbers of the reference terminal 7
Appendix 3: Annual berth production based on vessel arrival pattern 11
Appendix 4: Container storage capacity calculation 13
Appendix 5: Number of warehouse rows and stacking height 15
Appendix 6: Lane capacity calculations 17
Appendix 7: The rgv flows at the terminal 19
Appendix 8: Warehouse crane capacity calculation 27
Appendix 9: Warehouse crane occupancy rates 29
Appendix 10: Determining the number of rgv's at the terminal 33
Appendix 11: Flow chart of the simulation program 37
Appendix 12: The step by step trace of a rgv 41
Appendix 13: Input parameters for the experiment with one feeder vessel at Berth 1, unloading & Fast Loading 43
Appendix 14: Input parameters for the experiment with a feeder at Berth 1, loading containers from an adjacent berth 49
Appendix 15: Input parameters for the experiment "shuffling from Berth 1" 55
Appendix 16: Input parameters for the experiment with one mega vessel at Berth 2 unloading 59
Appendix 17: Input parameters for the experiment with one mega vessel at Berth 2, loading 65
Appendix 18: Input parameters for the experiment with one mega vessel at Berth 2 unloading and loading containers from an adjacent berth 67
Appendix 19: Input parameters for the experiment "shuffling from Berth 2" 75
Appendix 20: Input parameters for the experiment adjacent loading at all three berths 83
Appendix 21: Input parameters for the experiment adjacent loading at Berth 1 and 3 and unloading at Berth 2. 103
Appendix 22: Input parameters for the experiment adjacent loading on Berth 2, unloading on Berths 1 & 3 115
Appendix 23: Input parameters for the experiment "shuffling from 3 berths" 125
Appendix 24: Input parameters for the experiment adjacent loading at Berth 1 and 3 using 12 rgv's and unloading at Berth 2 133
Appendix 25: Input parameters for the experiment with disturbances in the portainer cycle times. 145
Appendix 26: Input parameters for the experiment with variable crane cycle times.. 151
Appendix 27: Table section 155
Appendix 28: Listing computer program .....
Appendix 29: Layout including rgv waiting spaces 160

Appendix 1: Lay out of the reference terminal

Appendix 2: Node numbers and stacknumbers of the reference terminal

Appendix 3: Annual berth production based on vessel arrival pattern

- *Calculation for mega vessels moored at Berth 2:*

Average portainer production:	60 moves/hour
Number of portainers on Berth 2:	6 portainers
TEU factor:	1.5
Operational days in a year:	365 days
Amount of TEU on a mega vessel:	8000 TEU
Desired berth occupancy rate:	50 %
Terminal share for unloading & loading:	60 %

Total amount of TEU per call: 8000* 0.6* 2 =9450 TEU

Handling hours: $\frac{9450}{60*6*1.5} = 17.5$ hours

Including maneuvering and mooring: 19.5 hours

Arrival scheme: One mega vessel every $\frac{19.5*365}{(365*24)*0.5} = 1.6$ days

Assume an arrival frequency of 1 mega vessel every 1.5 days

The berth occupancy rate will then amount to: $\frac{\left(\frac{365}{1.5}*19.5\right)}{365*24} = 54\%$

Total of handled TEU on Berth 2: $\frac{60*6*1.5*365*17.5}{1.5} = 2.3$ million TEU's/year

- *Calculation for feeder vessels moored at Berth 1 & 3*

Average portainer production:	60 moves/hour
Number of portainers on Berth 2:	4 portainers
TEU factor:	1.5
Days in a year:	365 days
Amount of TEU on a feeder vessel:	1575 TEU
Desired berth occupancy rate:	50 %
Terminal share for unloading & loading:	100 %
Total amount of TEU per call:	3150 TEU

Handling hours: $\frac{3150}{60*4*1.5} = 8.75$ hours

Including maneuvering and mooring: 10.75 hours

Arrival scheme per berth: One feeder vessel every $\frac{10.75*365}{(365*24)*0.5} = 0.9$ days

Assume an arrival frequency of 1 feeder vessel every day for each berth

The berth occupancy rate will then amount to: $\frac{\left(\frac{365}{0.5} * 10.75\right)}{365 * 24 * 2} = 44.8\%$

Total of handled TEU on Berth 1&3: $\frac{2 * 60 * 4 * 1.5 * 365 * 8.75}{1} = 2.3$ million TEU's/year

The total annual amount of handled TEU on Berth 2 is the same as the total annual amount of handled TEU on Berth 1 & 3. This of course is no coincidence. By carrying out an iteration process with the arrival frequency of the feeder vessels and their carrying capacities as variables the annual throughputs on Berth 2 and Berths 1 & 3 can be kept equal. This has resulted in a feeder vessel capacities of 1575 TEU.

Appendix 4: Container storage capacity calculation

The calculation of the storage capacity of the stack has been carried out based on the method as described in Chapter 3.7:

The used formula:

$C_0 = q_u + q_u(D_u - p)/D_u + q_u(D_u - 2p)/D_u + ... + q_u(D_u - np)/D_u$
$+ q_l + q_l(D_l - (p-1))/D_l + q_l(D_l - (2p-1))/D_l + ... + q_l(D_l - (np-1))/D_l$

With:
q_u = number of TEU to be unloaded from a mega vessel
q_l = number of TEU loaded onto a mega vessel
D_u = number of days that the feeder vessels are allowed to deliver containers in advance for the mega vessel
D_l = number of days that the feeder vessels are allowed to retrieve containers after the arrival of the mega vessel
p = periodic interval of mega vessels in days
n = 1, 2, 3, ...

Resulting in:

$$C_0 = 4800 + 4800 * \left(\frac{5-1.5}{5}\right) + 4800 * \left(\frac{5-3}{5}\right) + 4800 * \left(\frac{5-4.5}{5}\right)$$
$$+ 4800 + 4800\left(\frac{5-(1.5-1)}{5}\right) + 4800\left(\frac{5-(3-1)}{5}\right) + 4800\left(\frac{5-(4.5-1)}{5}\right) + 4800\left(\frac{5-(6-1)}{5}\right)$$

= 24,000 TEU

Appendix 5: Number of warehouse rows and stacking height

The number of warehouses and the stacking height is directly related to each other. Increasing the number of warehouses will subsequently decrease the stacking height, but will increase the number of required warehouse cranes, service points and terminal area. Between these relations an economic optimum should be calculated. In this study the terminal area will be normative.

The required stack capacity has been calculated to amount to 24,000 TEU in Chapter 3. The maximum stacking height has, quite arbitrary, been assumed to be able to amount to 11 containers or 12 stories in height. The following formula has been used.

$$S = \frac{C}{W * B * L}$$

S = stacking height
C = required stack capacity (= 24,000 TEU)
W = number of warehouse units (= 70)
B = number of TEU distributed over width of the warehouse (= 4)
L = number of TEU distributed over the length of the warehouse (= 8)

6 warehouse rows
Using 6 warehouse rows the total amount of warehouse units is: 84. This will amount to a stacking height of 8.9 containers. Including a small shuffling buffer, as containers are stored two units' deep storage height amounts to 9-10 containers.

5 warehouse rows
Using 5 warehouse rows the total amount of warehouse units is: 70. This will amount to a stacking height of 10.71 containers. Including a small shuffling buffer, as containers are stored two units' deep storage height amounts to 11-12 containers.

In this study the choice has been made to simulate the terminal with 5 warehouses resulting in a stacking height of 12 containers. As emphasis of this study has not been economic or constructive but a logistic one, a recommendation will be made for a further analysis on stacking height.

Appendix 6: Lane capacity calculations

The lane capacity can be calculated by determining the minimal accepted time between 2 succeeding rgv's driving on the same lane. This follow up time can be established by determining the time a rgv requires for entering a rgv lane.

Yard Lane capacity
The capacity of the yard lane is determined by the time it that takes for a stationary rgv to enter the yard lane and accelerate to the required 1 m/s. This process can be seen in Figure 6.1.

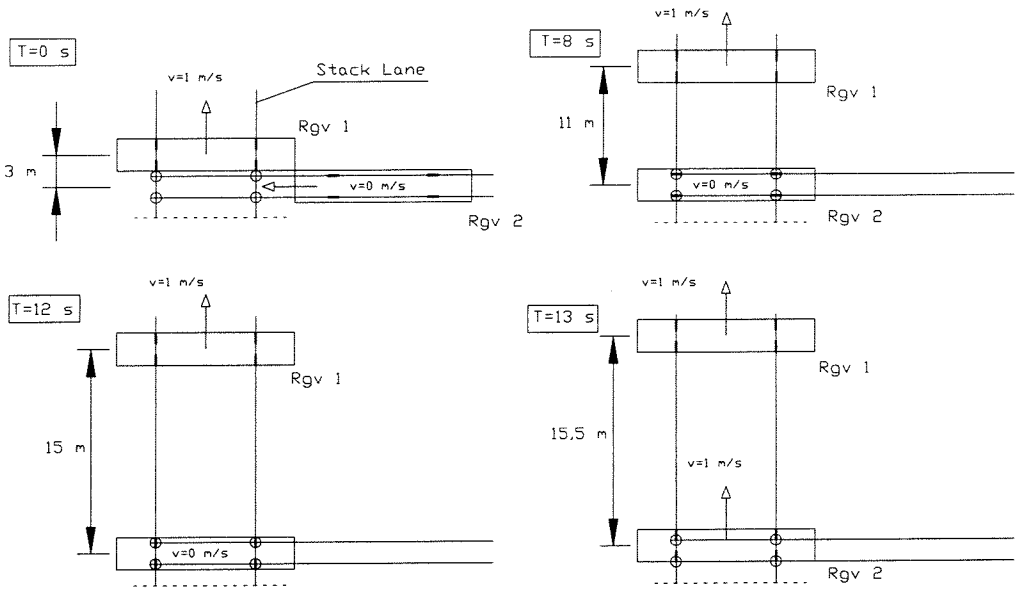


Figure 6.1 rgv entrance onto the Yard lane

At T=0 seconds the rgv is about to accelerate and decelerate from its position on for instance a warehouse lane onto the yard lane. Eight seconds later on T=8 s. the rgv has just decelerated to 0 m/s and is about to have its wheels turned 90°. After 4 seconds the wheels are turned and the rgv is about to accelerate to 1 m/s. This takes one second.
The yard lane capacity can now be determined by calculating the necessary follow up time between the hearts of 2 succeeding rgv's. The follow up time is: 15,5 seconds. The yard lane capacity thus is almost 4 rgv's per minute.

Inter-berth Lane capacity
The capacity of the Inter-berth lanes is determined in a similar fashion. By calculating the required time for a rgv to enter the inter-berth lane, have its wheels turned and accelerate to 3 m/s the follow up time between the hearts of two rgv's can be determined as can be seen in Figure 6.2.

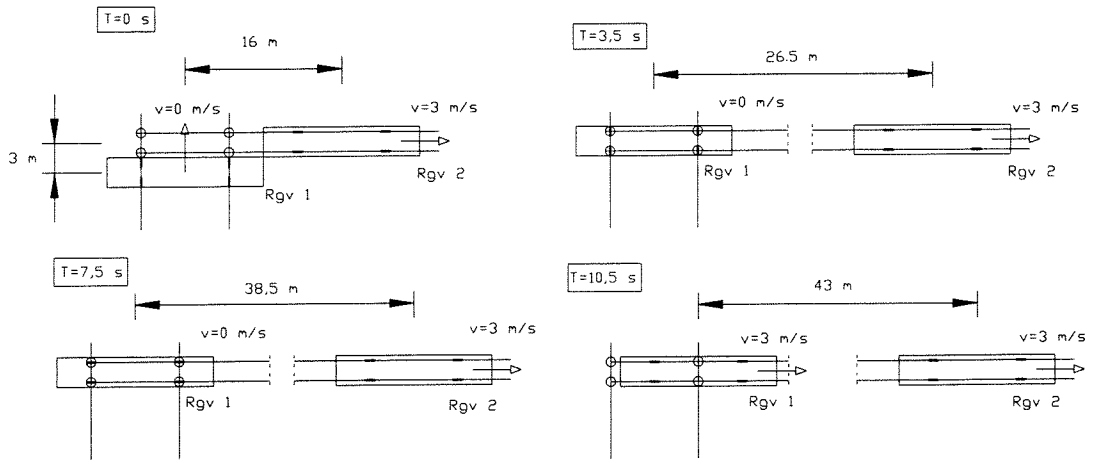


Figure 6.2., rgv entrance onto the inter-berth lane

Accelerating with 1 m/s^2 onto the inter-berth lane takes the rgv 3,5 seconds. The 90°-wheel rotation takes 4 seconds. Accelerating to 3 m/s will bring the total amount required time to 10,5 seconds. The heart to heart follow up time can be determined as 43 meters. Dividing this distance by the speed on the inter-berth lane comes down to a follow up time of 14,3 seconds, or about 4 rgv's per minute.

Appendix 7: The rgv flows at the terminal

Calculations concerning the expected rgv flows at the terminal.

After having calculated the capacities of the two different rgv lanes in Appendix 6, it is possible to analyse the different situations at the terminal and to judge whether the terminal can handle the situation in terms of capacity.
In this appendix the terminal is schematised; only the important nodes and lanes are shown and a few situations are considered. The terminal is shown in Figure 7.1. The nodes at the top are the nodes that connect the yard-lanes with the inter-berth lanes located behind the yards. The numbers refer to the node numbers.

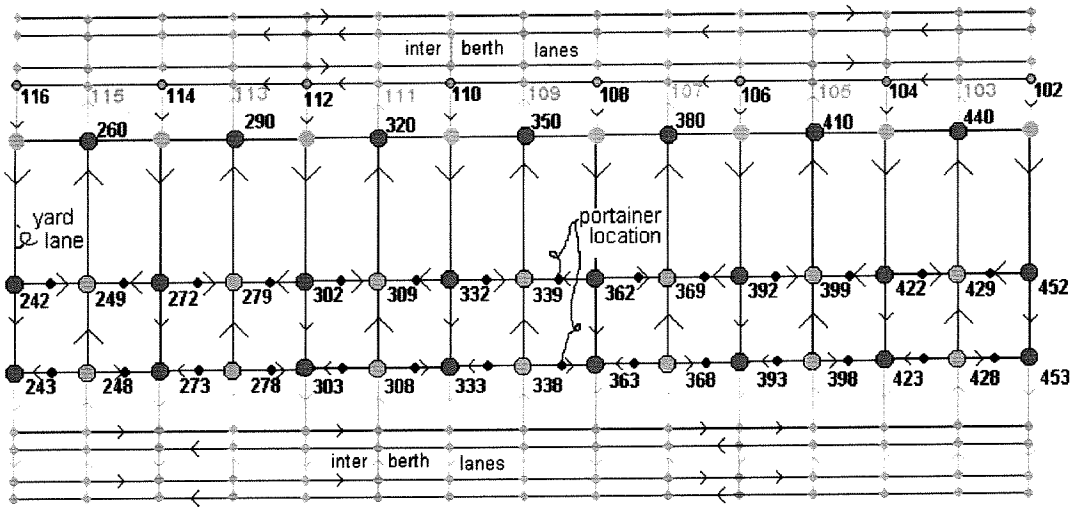


Figure 7.1, the terminal

The green nodes indicate an entrance onto the yard-lanes. At the red nodes the flow of rgv's can leave the yard lane and enter the inter-berth lanes located behind the yard. The individual warehouse lanes are not shown in this figure. The inter-berth lanes are connected to the yard-lanes by shortcuts. The black dots are the portainer locations where the portainers are situated, as can be seen in Figure 7.1, there are two ways to get to one portainer. In this appendix the node numbers are not important but the mean rgv flows are. Therefore the terminal is further schematised as can be seen in Figure 7.2.

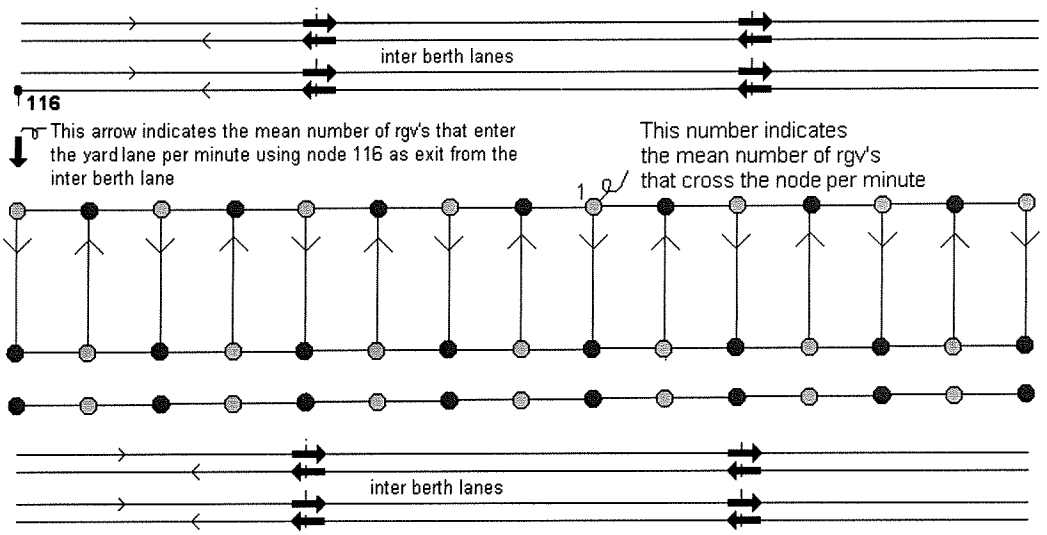


Figure 7.2 schematisation of the terminal

The black arrows refer to mean rgv flows.

Using this schematisation, four situations will be evaluated.

- One feeder vessel is moored at Berth 1 and is being unloaded.
- Three vessels are berthed. All of them are being loaded with containers from adjacent berths.
- Three vessels berthed, the two feeder vessels are being loaded with containers from adjacent berths and the mega vessel is being unloaded at Berth 2.
- Three vessels berthed, 1 feeder vessel is being unloaded at Berth 1, the 2 other vessels are being loaded at Berth 2 and 3 with containers from adjacent berths.

Situation I: one feeder vessel moored at Berth 1 during unloading.

The container flows during the unloading of a feeder vessel moored at Berth 1 are restricted to the berth where the vessel is moored. The flows are realised by the assumption that portainers have a mean portainer cycle time of 1 rgv per minute. Knowing that two portainers are connected to a yard-lane means 2 rgv's per minute will have to pass the node at the end of the portainer locations. This is shown in Figure 7.3.

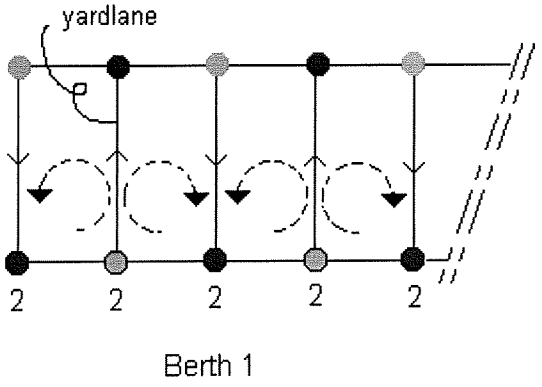


Figure 7.3, the mean rgv flows during the unloading of a feeder vessel moored at Berth 1.

This situation should cause no problem. The capacity of the yard lanes is about 4 rgv's per minute.

Situation II: three vessels moored while all are loading from adjacent berths

This situation is more complex than the previous one. First the needed rgv flows will be examined per individual vessel. At the end of this situation description they will be integrated, in order to get an overall picture of the total rgv flows.

Feeder vessel moored at Berth 1

The feeder vessel moored at Berth 1 is being loaded by rgv's receiving their containers from Yard 2. The rgv's will drive to Yard 2 using the two inter-berth lanes. In this case they will use the top inter-berth lanes and the bottom inter-berth lanes. With a mean portainer production of one move per minute this means the rgv flow on the top inter-berth lanes is 2 rgv moves per minute in both directions and the rgv flow on the bottom inter-berth lanes is 2 rgv moves per minute in both directions. The rgv's enter these lanes by using the red nodes. At the top of the yard two yard-lanes exit and connect to the inter-berth lanes. One rgv exits the yard-lane and enters the inter-berth lanes per minute. These are the red nodes (1/2*2).
The yard has three yard-lane exit possibilities for the rgv's to enter the bottom inter-berth lanes. The mean rgv flows per minute are shown in Figure 7.4.

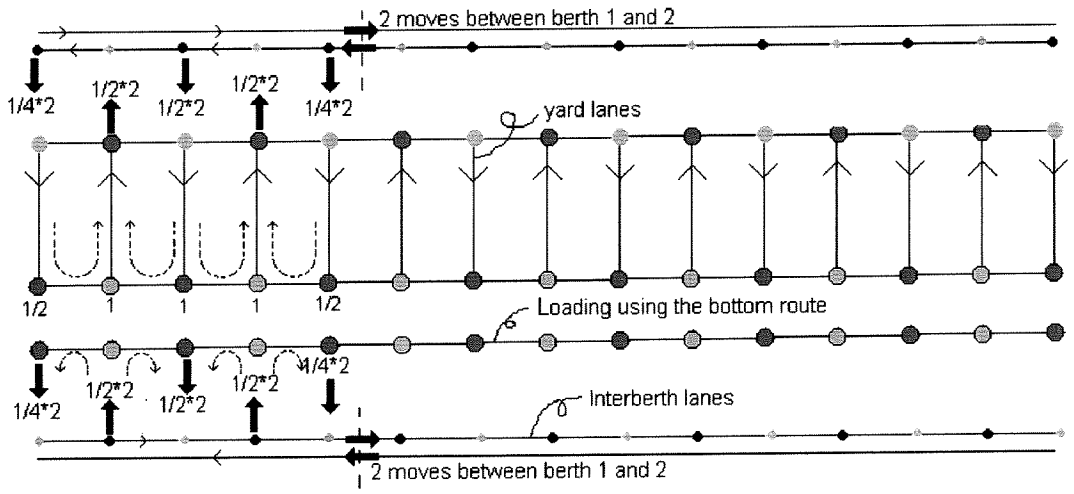


Figure 7.1, the mean flows at Berth 1 created by rgv's during adjacent loading at Berth 1

Mega vessel moored at Berth 2.

The mega vessel moored at Berth 2 is being loaded with containers retrieved from the adjacent berths, Berth 1 and 3. The flows look like the ones created by the rgv's working for the feeder berths. The flows are shown in Figure 7.7.

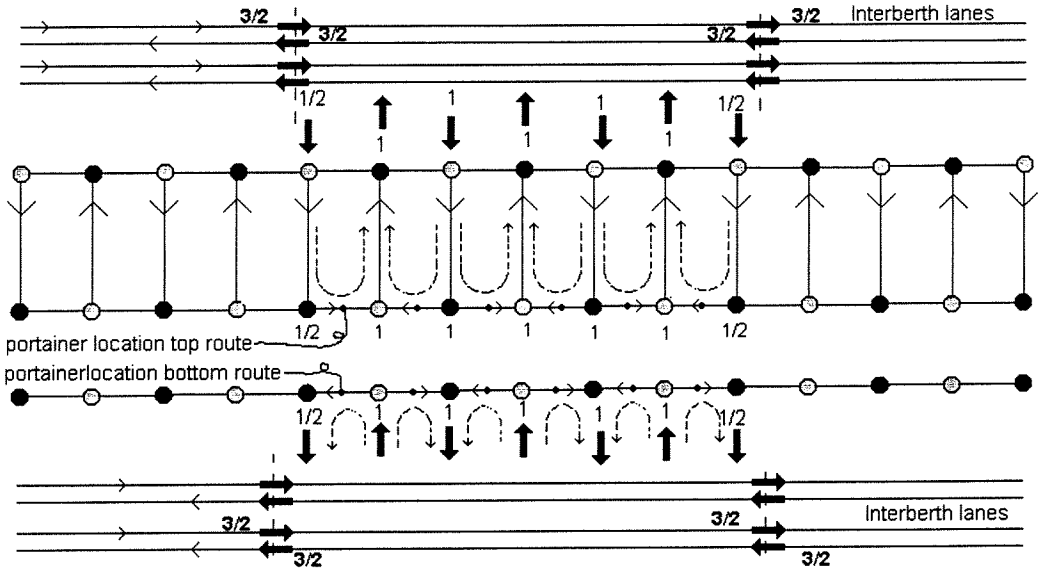


Figure 7.7 mean flows created at Berth 2 by rgv's loading at Berth 2

The rgv' s from Berth 2 retrieve their containers from Berth 1 and 3. This is shown in Figure 7.8.

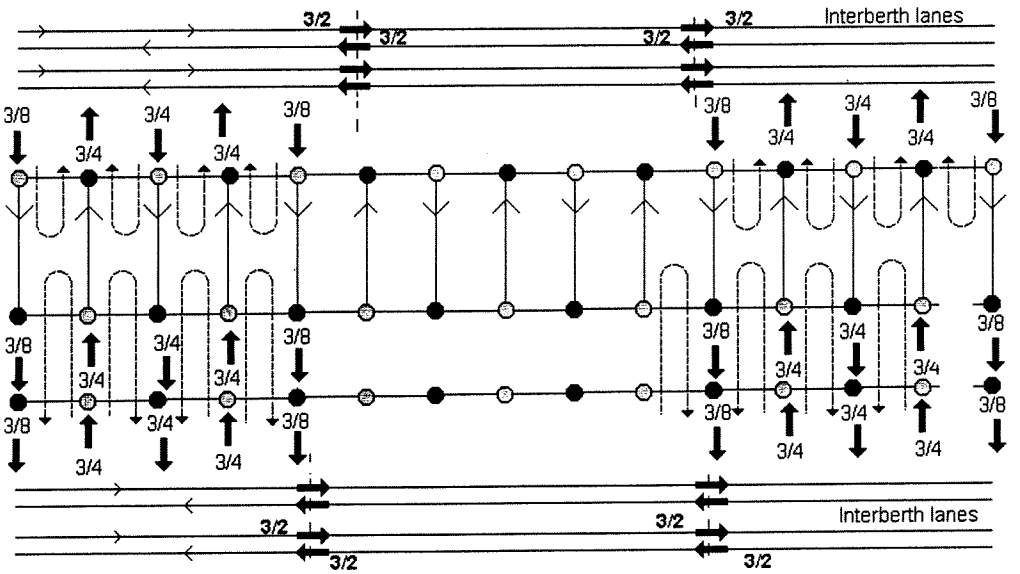


Figure 7.8, flows created by rgv's from Berth 2 collecting containers behind Berth 1 & 3

Integrating all the rgv flows discussed above gives a total rgv flow pattern at the terminal of the 3 vessels loading from adjacent berths. This total flow is shown in Figure 7.9.

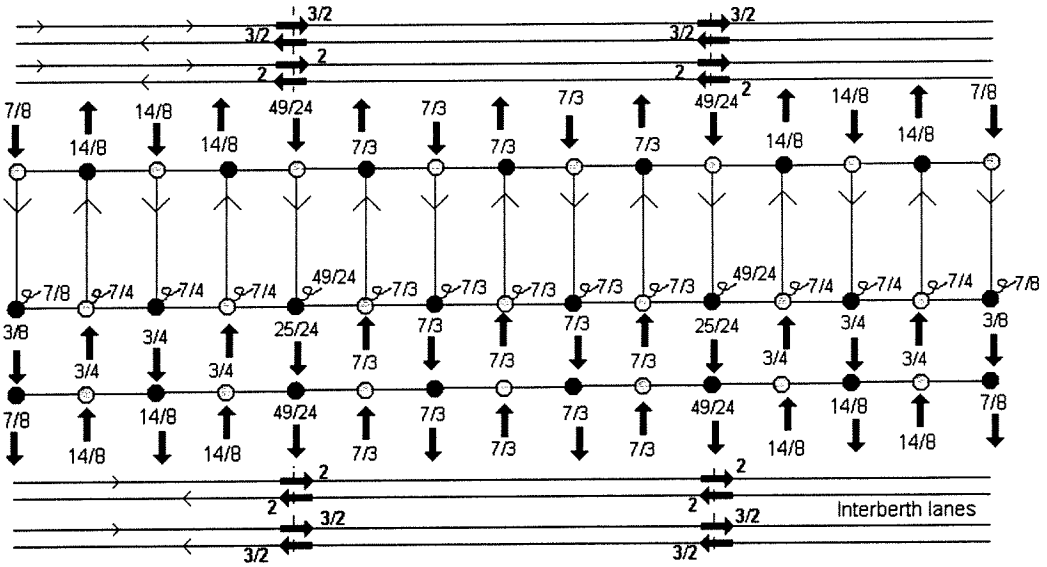


Figure 7.9, mean total flow of rgv's when 3 vessels are being loaded from adjacent berths

As the yard lanes and the inter-berth lanes have a lane capacity of about 4 rgv's per minute the designed layout should work.

Situation III: three vessels moored with two vessels loading from an adjacent berth and one unloading

In this case a mega vessel is being unloaded by its portainers by delivering the unloaded containers to the stacks in Yard 2. Feeder vessels moored at Beth 1 and Berth 3, are being adjacent loaded from the same yard: Yard 2. This combination will put great pressure on the yard-lanes of Yard 2. When a vessel is being handled, the portainers handling the vessel have a mean cycle of one container per minute. The total of container flows can be seen in Figure 7.10.

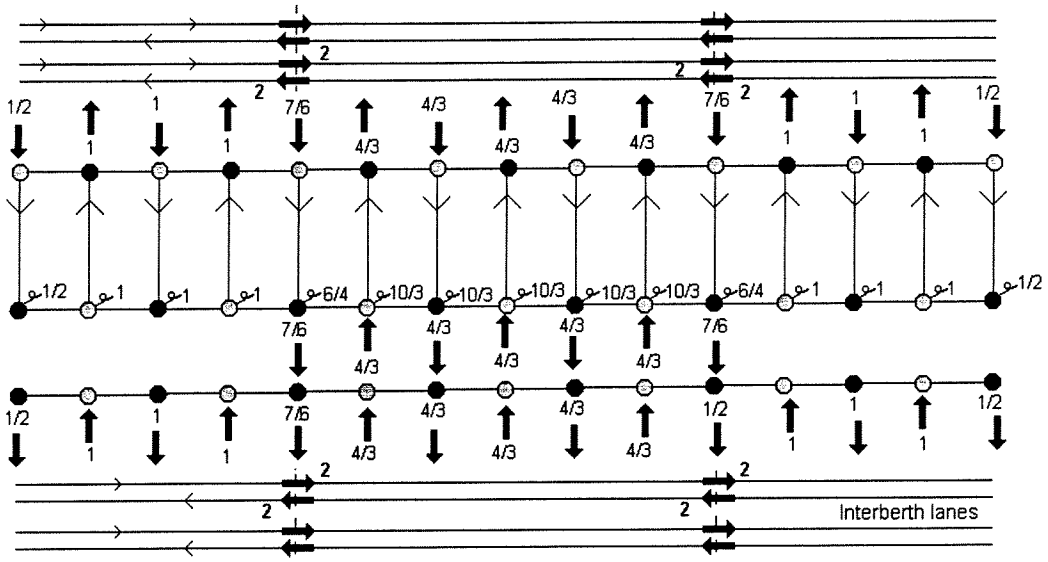


Figure 7.10, total flow when Berth 2 is being unloading and Berth 1&3 are being loaded from the adjacent Yard 2.

The nodes connecting the portainers to the yard-lanes are very busy and demand a capacity of 10/3 rgv per minute. The capacity of these lanes is about 4 rgv's per minute. The computer simulation will have to point out whether or not the terminal is capable of handling this many rgv's a minute.

In this situation there is the possibility to make more use of the top inter-berth lanes and less use of the bottom inter-berth lanes. This will relieve the pressure on the nodes behind the portainers. A balance could be found.

Situation IV: three vessels moored, with Beth 1 unloading and Berth 2 & 3 adjacent loading.

In this situation, three vessels are moored at the berths. The feeder vessel at Berth 1 is being unloaded and the mega vessel at Berth 2 and the feeder vessel at Berth 3 are being loaded with containers from the adjacent yards. The total flow of rgv's can be seen in Figure 7.11.

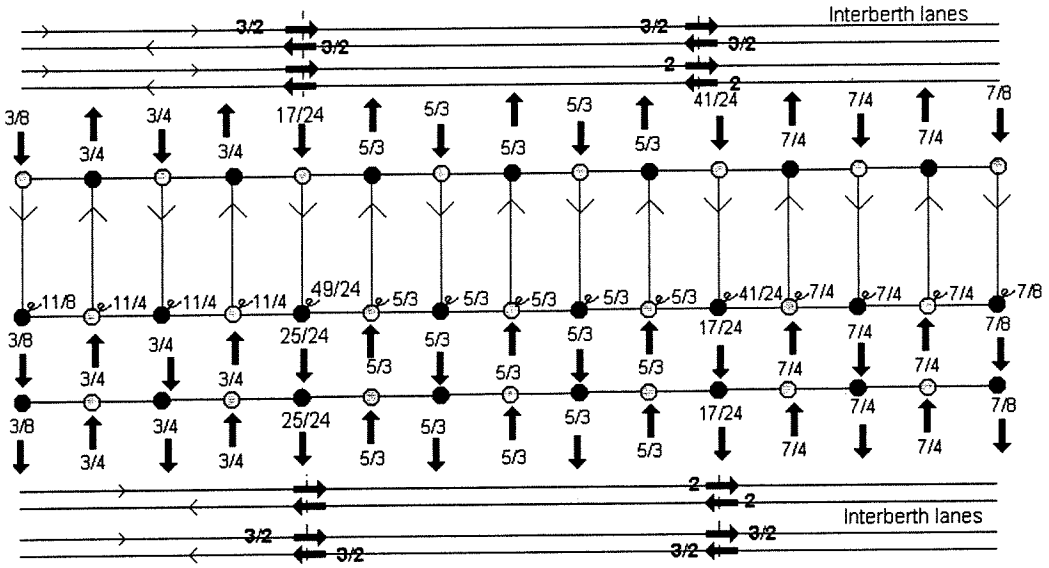


Figure 7.11, three vessels at the berth; unloading at Berth 1 adjacent loading at Berth 2 and 3.

Appendix 8: Warehouse crane capacity calculation

This calculation is made in order to get a first impression of the production capacity of the warehouse crane. Some assumptions have been made in order to simplify this calculation.

Assumptions

- Horizontal movements will take place during vertical transportation, implying no extra time will be taken in to account.
- Shuffling time will be left out of the calculation. If the terminal operator has in advance information on vessel schedules and vessel unloading and loading schemes, he can anticipate on his actions and minimise the amount of shuffling moves.
- The stacking height in the warehouse will be 12 containers high, making it a 13-storey warehouse when including the ground floor where the warehouse lanes and the service points are placed.
- The operating speeds of the warehouse cranes are assumed to have the following properties.
  - Crane travelling speed: 2.5 m/s
  - Lift speed:
    - Full up: 0.5 m/s
    - Empty up/down: 0.83 m/s
    - Full down: 0.83 m/s
  - Shuttle speed: 0.5 m/s
  - Acceleration: 0.5 m/s^2

Calculation

Maximum lifting height = Storey height * Number of stories = 3*11 = 33 m.

- Lifting up (full):
  - Acceleration time: $t_a = v/a = 0.5/0.5 = 1$ second.
  - Acceleration distance: $X_a = {}^1/_2 * a * t^2 = 0.25$ m. $\Rightarrow 2*0.25 = 0.5$ m.
  - Full speed lifting: 33 m – 0.5 m = 32.5 m $\Rightarrow t_v = 32.5/0.5 = 65$ seconds.
  - Total lifting time: $T_t = 65 + 2 = 67$ seconds.

- Lifting up (empty):
  - Acceleration time: $t_a = v/a = 0.83/0.5 = 1.66$ seconds.
  - Acceleration distance: $X_a = {}^1/_2 * a * t^2 = 0.69$ m. $\Rightarrow 2*0.69 = 1.38$ m.
  - Full speed lifting: 33 m – 1.38 m = 31.62 m $\Rightarrow t_v = 31.62/0.83 = 38.1$ seconds.
  - Total lifting time: $T_t = 38.1 + 3.3 = 41.4$ seconds.

- Lowering takes the same amount of time as lifting up empty $\Rightarrow$ 41.4 seconds.

Shuttle:

- 1 rack deep (2.6 m).
  - Acceleration time: $t_a = v/a = 0.5/0.25 = 2$ seconds.
  - Acceleration distance: $X_a = {}^1/_2 * a * t^2 = 1$ m. $\Rightarrow 2*1 = 2$ m.
  - Full speed lifting: 2.6 m – 2 m = 0.6 m $\Rightarrow t_v = 0.6/0.5 = 1.2$ s
  - Total operating time: $T_t = 1.2 + 2 + 2*2$ (jacking up & down) = 5.2 seconds.

- 2 racks deep (5.8 m).
  - Acceleration time: $t_a = v/a = 0.5/0.25 = 2$ seconds.
  - Acceleration distance: $X_a = {}^1/_2 * a * t^2 = 1$ m. $\Rightarrow 2*1 = 2$ m.
  - Full speed lifting: 5.8 m – 2 m = 3.8 m $\Rightarrow t_v = 3.8/0.5 = 7.6$ s
  - Total operating time: $T_t = 7.6 + 2 + 2*2$ (jacking up & down) = 13.6 seconds.

Situation 1: Retrieving a container out of a rack:
Actions:
1. Lifting up to a rack.
2. Loading container onto the crane with the shuttle.
3. Lowering the crane.
4. Unloading the container onto the service point.

2 Deep:

	12th storey (max. height)	6th storey (halfway)	1st storey (lowest)
1.	41.4	23.3	5.3
2.	23.2	23.2	23.2
3.	41.4	23.3	5.3
4.	10.4	10.4	10.4
Total:	116.4 s	80.2 s	44.2 s

1 Deep:

Total:	103.6 s	67.4 s	31.4 s

Average Time for situation 1 = 73.9 s

Situation 2: Depositing a container in a rack:
Actions:
1. Loading container from a service point onto the crane with the shuttle.
2. Lifting up to a rack.
3. Unloading the container into the rack.
4. Lowering the crane.

2 Deep:

	12th storey (max. height)	6th storey (halfway)	1st storey (lowest)
1.	10.4	10.4	10.4
2.	67	37	7
3.	23.2	23.2	23.2
4.	41.4	23.3	5.3
Total:	142 s	93.9 s	45.9 s

1 Deep:

Total:	131.2 s	81.1 s	33.1 s

Average Time for situation 2 = 87.9 s

Total average time without shuffling: (77.1 + 89.9)/2 = 80.9 s

Evaluation

The calculated maximum capacity of the yard-lane can amount to 240 moves/hour. Assuming the rgv flow is evenly spread out over each warehouse-lane implicates that each lane must be able to cope with 240/10 = 24 moves an hour, or one move every 2.5 minutes or one move every 150 seconds.

Subtracting the two averages gives a buffer of 150.0 – 80.9 = 69.1 seconds. It seems one warehouse crane per warehouse-lane should be sufficient, bearing in mind that these calculations assume that the maximum rgv flow through the yard-lane is evenly spread out over all 10 connecting warehouse lanes. The computer simulation must give a clarification on this assumption.

Appendix 9: Warehouse crane occupancy rates

Unloading/fast-loading of a container vessel.

Based on the assumed 1.5 TEU ratio, 50% of the handled containers will be stored in the 20' container stack while the remaining 50 % will be stored in the 40' container stack. Separate warehouse crane capacity calculations have been made for the 20' and 40' container stack. As the portainer production has been determined to achieve 1 container per minute, both 20' and 40' container stacks will receive an average of one container per 30 seconds.

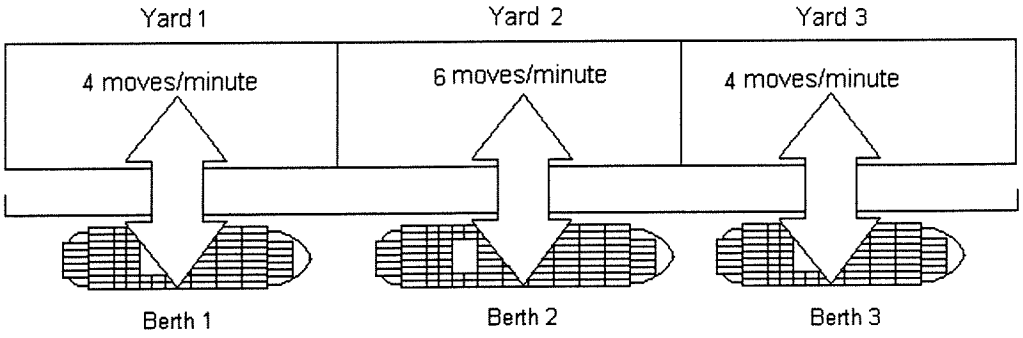


Figure 9.1, unloading/fast-loading on the terminal

Occupancy rate calculation of the 20' container warehouse crane

The 20' container stack located behind an unloading/fast-loading portainer receives a container flow of 30 containers per hour.
A stack has the capacity of one move per 80 seconds (Appendix 8) or 45 moves per hour.
The 20' container stack contains 2 warehouse cranes.

The average warehouse crane occupancy rate is: $\left(\frac{30}{45*2}\right)*100\% = 33.3\ \%$

Occupancy rate calculation of the 40' container warehouse crane

The 40' container stack located behind an unloading/fast-loading portainer also receives a container flow of 30 containers per hour.
A stack has the same capacity of one move per 80 seconds or 45 moves per hour.
The 40' container stack contains 3 warehouse cranes.

The average warehouse crane occupancy rate is: $\left(\frac{30}{45*3}\right)*100\% = 22.2\ \%$

Unloading/fast-loading of a container vessel moored at Berth 1 and adjacent loading of a mega vessel moored at Berth 2.

Added to the 30 containers an hour that a warehouse crane receives during unloading/fast-loading of a vessel moored at Berth 1, are the empty rgv's that arrive from Berth 2 in order to retrieve a container at Stack 1.
The additional number of rgv's is based on the portainer production of Berth 2 being 60 moves per portainer per hour or 360 moves from Berth 2 per hour. As the rgv's retrieve their containers from Berth 1 and Berth 3, 180 moves per hour will be the average number of visiting rgv's from Berth 2. Divided over the four portainers located on Berth 1, results in 45 rgv's for both 20' warehouses and 40' warehouses.

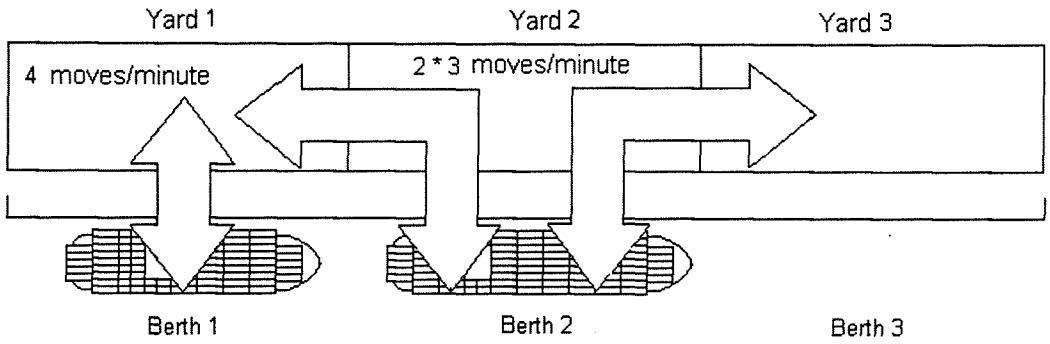


Figure 9.2, unloading/fast-loading on Berth 1 and adjacent loading on Berth 2

Occupancy rate calculation of the 20' container warehouse crane

The 20' container stack located behind an unloading/fast-loading portainer receives a container flow from Berth 2 of 22.5 empty rgv's per hour.

The average warehouse crane occupancy rate is: $33.3\ \% + \left(\frac{22.5}{45 * 2}\right) * 100\% = 58{,}3\%$

Occupancy rate calculation of the 40' container warehouse crane

The 20' container stack located behind an unloading/fast-loading portainer receives a container flow from Berth 2 of 22.5 empty rgv's per hour

The average warehouse crane occupancy rate is: $22.2\ \% + \left(\frac{22.5}{45 * 3}\right) * 100\% = 39\%$

Unloading/fast-loading of a mega container vessel moored at Berth 2 and adjacent loading of two feeder vessels moored at Berth 1 and Berth 3.

The third situation that may occur on the terminal is when a mega vessel is unloading/fast-loading at Berth 2 and two feeder vessels are adjacent loading on the Berths 1 and 3. The warehouse cranes located in the stack behind Berth 2 will have the highest occupancy rates that can be achieved on this terminal.
The basic occupancy rates of the warehouse cranes located in the stack behind Beth 2 is 22.2 % for the 20' stack and 33.3 % for the 40' stack as has been calculated before. Adding to this the adjacent loading rgv's from Berth 1 and Berth 3 will increase the occupancy rates of the warehouse cranes.

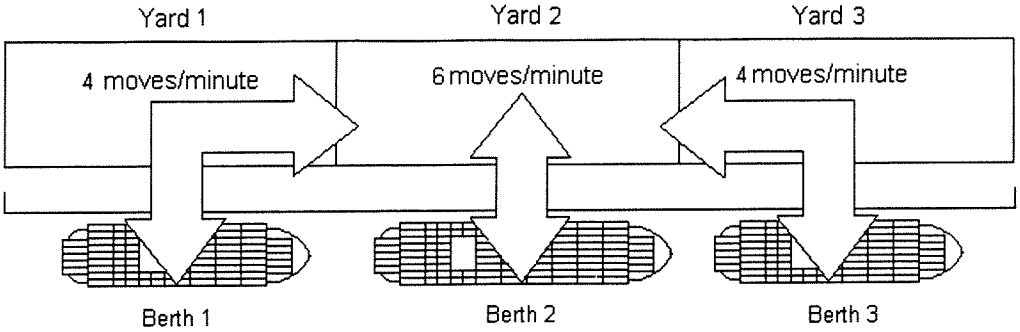


Figure 9.3, adjacent loading on Berths 1 and 3 and unloading/fast-loading on Berth 2

Occupancy rate calculation of the 20' container warehouse crane

The 20' container stack located behind the unloading/fast-loading portainers of Berth 2 receives a flow adjacent loading rgv's from Berth 1 and 3 of 40 empty rgv's per hour.

The average warehouse crane occupancy rate is: $33.3\ \% + \left(\frac{40}{45 * 2}\right) * 100\% = 77{,}8\%$

Occupancy rate calculation of the 40' container warehouse crane

The 40' container stack located behind the unloading/fast-loading portainers of Berth 2 receives a flow adjacent loading rgv's from Berth 1 and 3 of 40 empty rgv's per hour.

The average warehouse crane occupancy rate is: $22.2\ \% + \left(\frac{40}{45 * 3}\right) * 100\% = 51{,}8\%$

Appendix 10: Determining the number of rgv's at the terminal

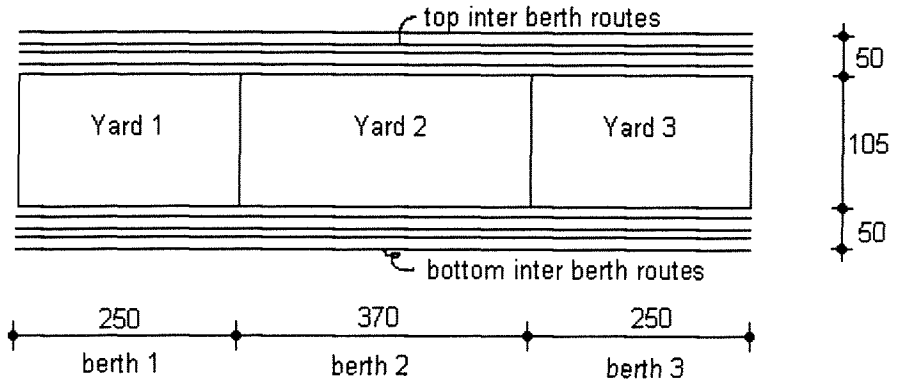


Figure 10.1 schematisation of the terminal

Figure 10.1 is a schematisation of the terminal with some dimensions. This figure makes it possible to determine the number of rgv's per portainer needed for the different procedures at the terminal. The procedures are unloading a vessel, loading a vessel and shuffling at the terminal.

Number of rgv's needed per portainer when loading a vessel from an adjacent berth.

When a feeder vessel is being loaded at Berth 1 with containers retrieved from the yard of Berth 2 there are two routes for the rgv's to get to the Yard 2. They can take the top inter-berth route or the bottom inter-berth route of the terminal. The mean transportation cycle distance and needed time for the rgv's to cover this distance are calculated.

The cycle time for a rgv taking the top route from Berth 1 to 2 starting under the portainer at Berth 1 is shown in Table 10.1. In this example the mean distances have been used.

Direction	**distance[m]**	**speed [m/s]**	**time [s]**
The rgv has to go to the top inter berth route	110	1	110
The rgv has to go to the other berth	310	3	103
The rgv has to go to the stack entrance	60	1	60
The rgv drives through the stack	60	1	60
The rgv gets a container			15
The rgv has to go to the top inter berth route	60	1	60
The has to go back to its berth	310	3	103
The goes to portainer to deliver the container	110	1	110
The rgv is being served by the portainer			15
Total	1010		**636**

Table 10.1, the mean rgv cycle time to retrieve a container at Yard 2 for Berth 1, top route.

With a mean portainer cycle time of 60 seconds, the required number of needed rgv's is 636 [sec] / 60 [sec] or 10,5 rgv's per portainer.

The cycle time of a rgv taking the bottom route from Berth 1 to 2 and back starting under the portainer at Berth 1 is shown in Table 10.2.

Direction	distance[m]	speed [m/s]	time [s]
The rgv goes to the bottom inter berth route	10	1	10
The rgv has to go to the other berth	310	3	103
The rgv has to go to the stack entrance	60	1	60
The rgv drives through the stack	60	1	60
The rgv gets a container			15
The rgv goes to the bottom inter berth route	60	1	60
The has to go back to its berth	310	3	103
The goes to portainer to deliver the container	10	1	10
Total	820		**421**

Table 10.2, the mean rgv cycle time to retrieve a container at Yard 2 for Berth 1, bottom route.

The number of rgv's needed would be 421 [sec] / 60 [sec] is 7 rgv's per portainer.

The number of rgv needed per portainer when unloading a vessel

During the unloading of a vessel a rgv drives from the portainer to a stack in the stack-cluster behind this portainer, drops its container and drives back to the portainer in order to get the next container. These actions and their needed time demands, are shown in Table 10.3. The rgv starts at Berth 1 and is about to leave the portainer.

Direction	distance[m]	Speed [m/s]	time [s]
The rgv drives to a stack behind the portainer	55	1	55
The rgv drives through the stack	60	1	60
The rgv drops a container at a service point			15
The rgv drives to the portainer	55	1	55
The rgv gets a container from the portainer			15
The rgv drives under the portainer	60	1	60
Total	208		**248**

Table 10.3, the mean rgv cycle time when unloading a vessel

The number of rgv's needed would be 248 [sec] / 60 [sec] is 4.1 rgv's per portainer.

The number of needed rgv's when shuffling

During shuffling on the terminal the rgv's drive between the stacks using inter-berth lanes and yard-lanes. The actions during shuffling and their needed times are shown in Table 10.4. Assumed is a rgv starting behind Berth 1 and about to leave the stack to take the top route to the adjacent yard.

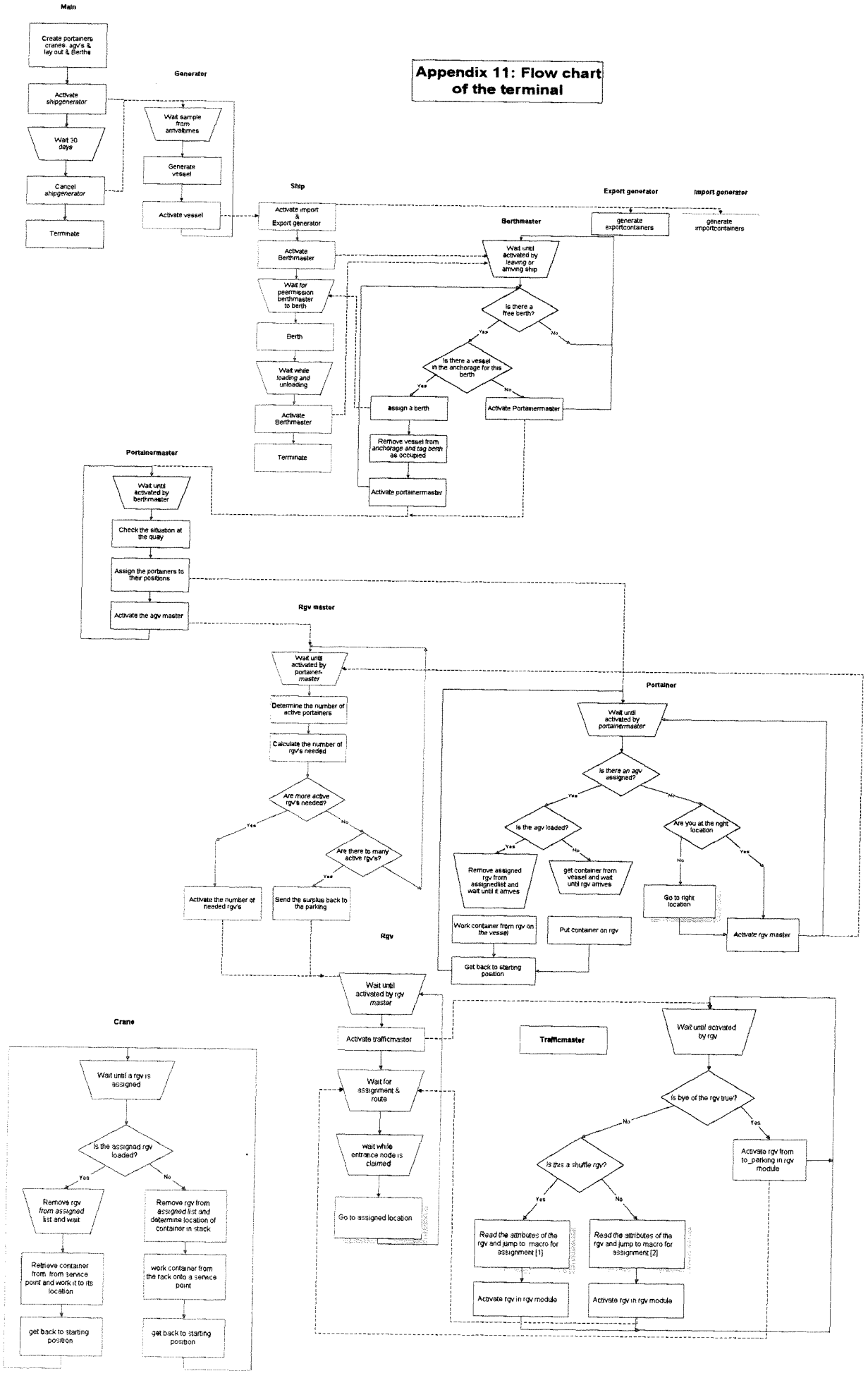
Direction	**distance[m]**	**speed[m/s]**	**time[s]**
The rgv drives to the top route	55	1	55
The rgv has to go to the other berth	310	3	103
The rgv has to go to a stack entrance	55	1	55
The rgv drives through the stack	60	1	60
The rgv drops a container in the stack			15
The rgv receives a container in the stack			15
The rgv drives to the top route	55	1	55
The rgv drives back to its berth	310	3	103
The rgv drives through the stack	60	1	60
The rgv drops a container at a service point			15
The rgv receives a container at a service point			15
Total			**551**

Table 10.4, The mean rgv cycle time when shuffling.

The cycle time of a rgv is 551 seconds. The crane capacity of the 5 warehouse cranes behind one portainer location is:

- The time needed for one crane to unload and load a rgv is 160 seconds.
- Per minute this results in 60/160 =0.375 rgv. With a total of 5 cranes behind a portainer location the mean capacity is 5*0,375=1,875 rgv per minute.
- If an occupancy rate of 50% is desired for the stacking cranes, this would result in 0,5*1,875=0,93 rgv per minute.
- With a rgv cycle time of 551 seconds the required number of rgv's is 0,9375*(551/60) = 8,6 rgv per portainer location when shuffling.

Appendix 11: Flow chart of the terminal



Appendix 12: The step by step trace of a rgv

In this appendix a rgv will be followed during its ride at the terminal. This appendix is realised by stepping through the program using one rgv per portainer while loading a vessel. This rgv starts in stack 62 and is working at berth 2 where a vessel is being unloaded. The entrance node of the rgv is node 292 and the exit node of the rgv is node 302. The rgv is about to leave the stack and is activated in the agvmod from toquay, line 80. In the table below the actions of the rgv, the time and the distances of the different actions are given. Column 3 refers to the module and the line in the module where the rgv gets it assignment. Column 5 in the table refers to the cur_node, an attribute of the rgv. Looking at this column the route of the rgv can be followed.

time[s]	rgv action	module/line	ΔT[s]	length[m]	cur_node
0	work exittime of cur_node	rgv/91	8	0	292
8	work discrotation	rgv/103	4	0	292
12	work nodedistance3/nodespeed	rgv/104	0	0	292
12	work nodedistance1/nodespeed	rgv/119	0,5	0,5	292
12.5	work max(0, (s_length of.....	rgv/121	2	2	292
14.5	work nodedistance2/nodespeed	rgv/125	0,5	0,5	293
15	work nodedistance3/nodespeed	rgv/127	0	0	293
15	work nodedistance1/nodespeed	rgv/119	0,5	0,5	293
15.5	work max(0, (s_length of.....	rgv/121	11	11	293
26.5	work nodedistance2/nodespeed	rgv/125	0,5	0,5	294
27	work nodedistance3/nodespeed	rgv/127	0	0	294
27	work nodedistance1/nodespeed	rgv/119	0,5	0,5	294
27.5	work max(0, (s_length of.....	rgv/121	2	2	294
29.5	work nodedistance2/nodespeed	rgv/125	0,5	0,5	295
30	work nodedistance3/nodespeed	rgv/127	0	0	295
30	work nodedistance1/nodespeed	rgv/119	0,5	0,5	295
30.5	work max(0, (s_length of.....	rgv/121	11	11	295
41.5	work nodedistance2/nodespeed	rgv/125	0,5	0,5	296
42	work nodedistance3/nodespeed	rgv/127	0	0	296
42	work nodedistance1/nodespeed	rgv/119	0,5	0,5	296
42.5	work max(0, (s_length of.....	rgv/121	2	2	296
44.5	work nodedistance2/nodespeed	rgv/125	0,5	0,5	297
45	work nodedistance3/nodespeed	rgv/127	0	0	297
45	work nodedistance1/nodespeed	rgv/119	0,5	0,5	297
45.5	work max(0, (s_length of.....	rgv/121	11	11	297
56.5	work nodedistance2/nodespeed	rgv/125	0,5	0,5	298
57	work nodedistance3/nodespeed	rgv/127	0	0	298
57	work nodedistance1/nodespeed	rgv/119	0,5	0,5	298
57.5	work max(0, (s_length of.....	rgv/121	2	2	298
59.5	work nodedistance2/nodespeed	rgv/125	0,5	0,5	299
60	work nodedistance3/nodespeed	rgv/127	0	0	299
60	work nodedistance1/nodespeed	rgv/119	0,5	0,5	299
60.5	work max(0, (s_length of.....	rgv/121	11	11	299

time[s]	rgv action	module/line	ΔT[s]	length[m]	cur_node
96.5	work nodedistance2/nodespeed	rgv/125	0,5	0,5	300
97	work nodedistance3/nodespeed	rgv/127	0	0	300
97	work nodedistance1/nodespeed	rgv/119	0,5	0,5	300
97.5	work max(0, (s_length of.....	rgv/121	7	7	300
104.5	work nodedistance2/nodespeed	rgv/125	0,5	0,5	301
105	work nodedistance3/nodespeed	rgv/127	0	0	301
105	work nodedistance1/nodespeed	rgv/119	0,5	0,5	301
105.5	work max(0, (s_length of.....	rgv/121	7	7	301
112.5	work nodedistance2/nodespeed	rgv/125	0,5	0,5	302
113	work nodedistance3/nodespeed	rgv/127	0	0	302
113	work nodedistance1/nodespeed	macro stat./5	0,5	0,5	302
113.5	work discrotation	macro stat./6	4	0	302
117.5	work exittime of cur_node	macro stat./16	8	0	302
125.5	work s_lenght of cur_section	macro stat./25	19	19	309
144.5					

In this table the loop the rgv takes in the rgv module becomes clear. This loop is line 109 to line 136 in the rgv module. The rgv gets out of the loop at the moment it reaches its exit node. In this case the exit node is the entrance of the portainer.
After 113 seconds the rgv leaves the yard lane and enters the portainer location. Thereafter it drives to the portainer and after 144 seconds the rgv is under the portainer and ready to receive a container.

The driving of the rgv from node 292 to the portainer is shown in Figure 12.1.

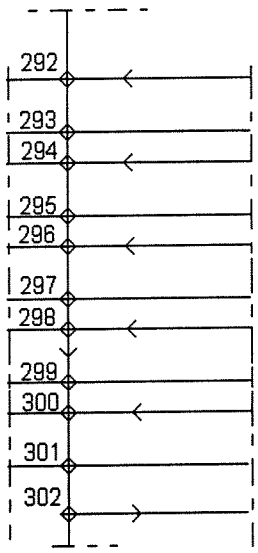


Figure 12.1, the route of the rgv from the stack to the portainer

Appendix 13: Input parameters for the experiment with one feeder vessel at Berth 1, unloading & Fast Loading

The basic parameters used for the experiment with the feeder vessel moored at Berth 1 during unloading are shown in Table 13.1. There are 1050 containers unloaded from the vessel. The parameters that have changed during the run are shown in Table 13.2. Inplace counter 1 refers to the number of containers in Yard 1 destined to be taken away by a vessel at Berth 1. These containers can be loaded fast.

Parameter	Value	
frombtob[1,2]	0	containers
frombtob[3,2]	0	containers
frombtob[2,1]	0	containers
frombtob[2,3]	0	containers
inplacecounter 1	1200	containers
inplacecounter 2	0	containers
inplacecounter 3	0	containers
seed of uniform	67	-
seed of arrivaltimes feeders	17	-
seed of arrivaltimes big	27	-
import generator feeders	1050	containers
export generator feeders	1050	containers
import generator big vessels	0	containers
export generator big vessels	0	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
rgv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of warehouse cranes at terminal	70	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	58	seconds

Table 13.1, Input parameters for experiment: feeder vessel at Berth 1

Parameter	Value	
frombtob[1,2]	1050	containers
Inplacecounter 1	150	containers

Table 13.2, output after experiment: feeder vessel at Berth 1 unloading

In Table 13.2 the parameter frombtob[1,2] indicates 1050 containers are unloaded from the vessel and stacked in Yard 1, these containers are destined for a vessel at Berth 2. Inplace counter 1 = 150 indicates 150 containers are left in Yard 1 destined to be taken away by a vessel from Berth 1. From the 1200 original containers in Yard 1, 1050 are taken away by the vessel.

13.1.1 Warehouse crane capacity calculation: The unloading of a feeder vessel

A feeder vessel moored at Berth 1 will have its containers unloaded in Yard 1. The number of unloaded containers is 1050. Half of the 1050 containers will be delivered to the 40'stacks and the other half to the 20' stacks.

- Required 20' warehouse crane operational time: 1050/2 * 80 seconds = 42,000 seconds
- Required 40' warehouse crane operational time: 1050/2 * 80 seconds = 42,000 seconds

Number of warehouse cranes in Yard 1:
- Number of warehouse cranes in the 20' stacks in Yard 1 = 4 * 2 = 8 warehouse cranes
- Number of warehouse cranes in the 40' stacks in Yard 1 = 4 * 3 = 12 warehouse cranes

Needed operational time per warehouse crane:

- $42{,}000/8 = 5{,}250$ seconds per 20' stack warehouse crane
- $42{,}000/12 = 3{,}500$ seconds per 40' stack warehouse crane

Simulated unloading time for the feeder vessel: 4.4 hours ⇒ 15,840 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(5{,}250/15{,}840\right) * 100\% = 33.1\%$

40' stack: $\left(3{,}500/15{,}840\right) * 100\% = 22.1\%$

13.1.2 Warehouse crane capacity calculation: The fast loading of a feeder vessel

A feeder vessel moored at Berth 1 will have its containers fast loaded from Yard 1. The number of to be loaded containers is 1050. Considering that the number of containers that have been retrieved from the 20' stacks and the 40' stacks are the same means that 525 containers will be delivered to both stack types.

- Required 20' warehouse crane operational time: 525 * 80 seconds = 42,000 seconds
- Required 40' warehouse crane operational time: 525 * 80 seconds = 42,000 seconds

Number of warehouse cranes in Yard 1:
- Number of warehouse cranes in the 20' stacks in Yard 1 = 4 * 2 = 8 warehouse cranes
- Number of warehouse cranes in the 40' stacks in Yard 1 = 4 * 3 = 12 warehouse cranes

Needed operational time per warehouse crane:

- $42{,}000/8 = 5{,}250$ seconds per 20' stack warehouse crane
- $42{,}000/12 = 3{,}500$ seconds per 40' stack warehouse crane

Required unloading time for the feeder vessel: 4.3 hours ⇒ 15,480 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(5{,}250/15{,}480\right) * 100\% = 33.9\,\%$

40' stack: $\left(3{,}500/15{,}480\right) * 100\% = 22.6\,\%$

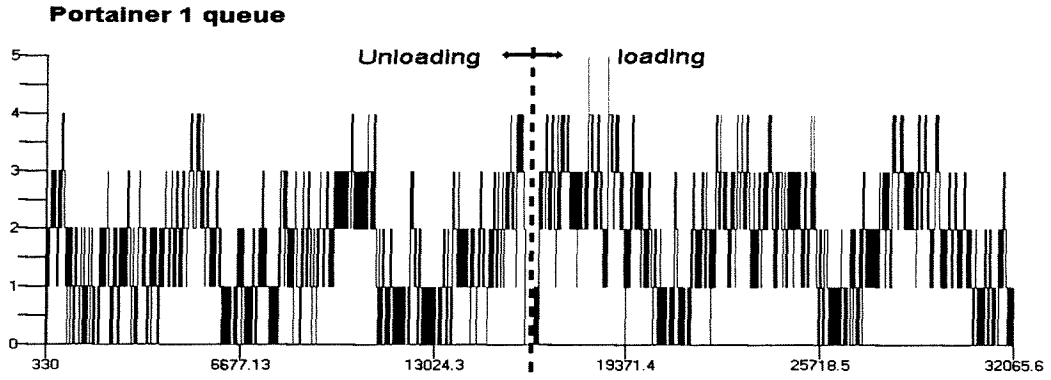


Figure 13.1, rgvqueue at portlocation 1 when unloading and loading a feeder fast

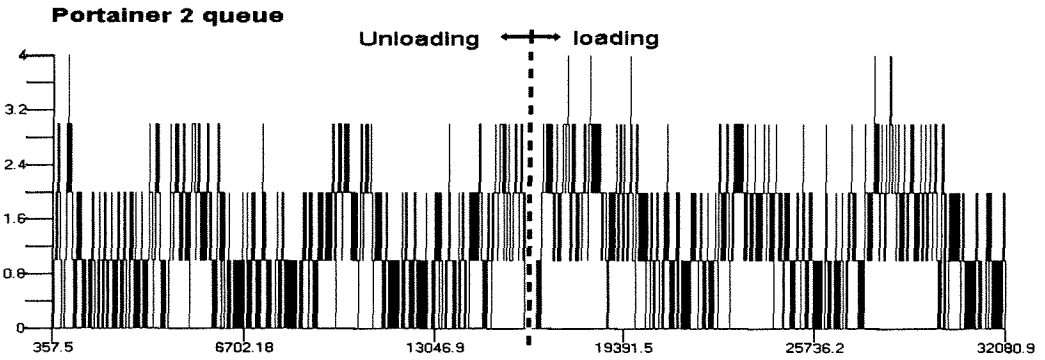


Figure 13.2, rgvqueue at portlocation 2 when unloading and loading a feeder fast

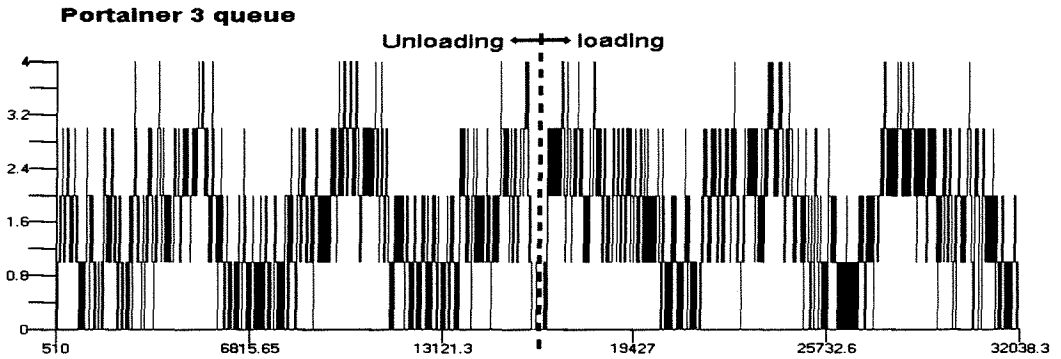


Figure 13.3, rgvqueue at portlocation 3 when unloading and loading a feeder fast

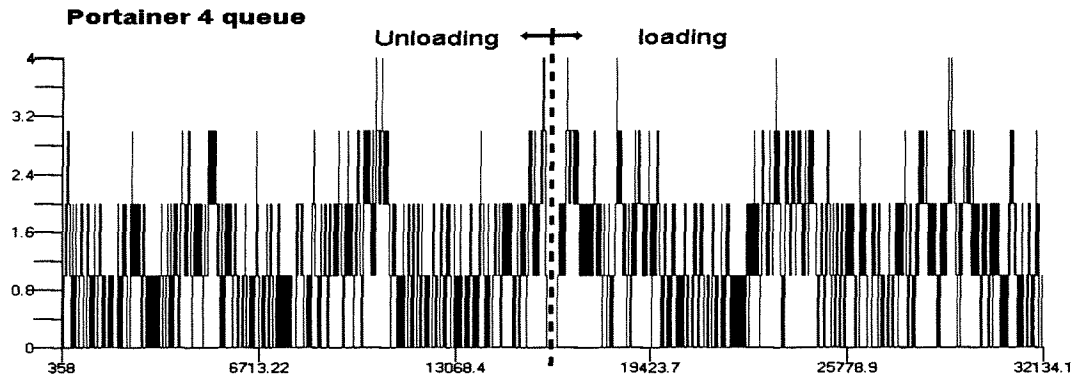


Figure 13.4, rgvqueue at portlocation 4 when unloading and loading a feeder fast

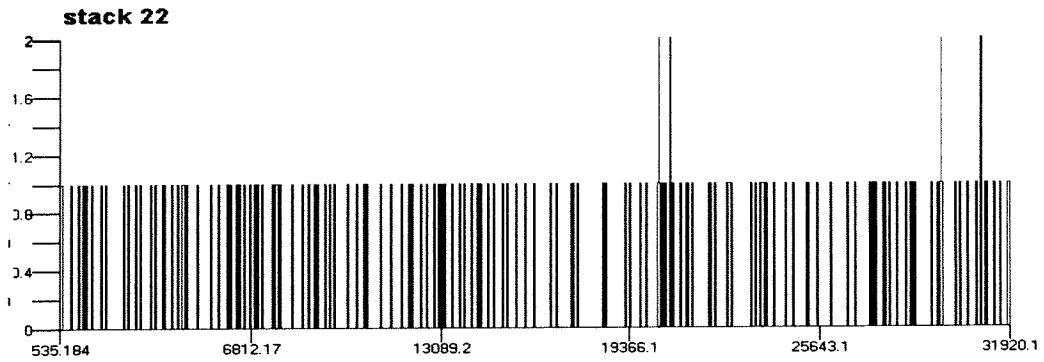


Figure 13.5, rgv queue on warehouse lane 22

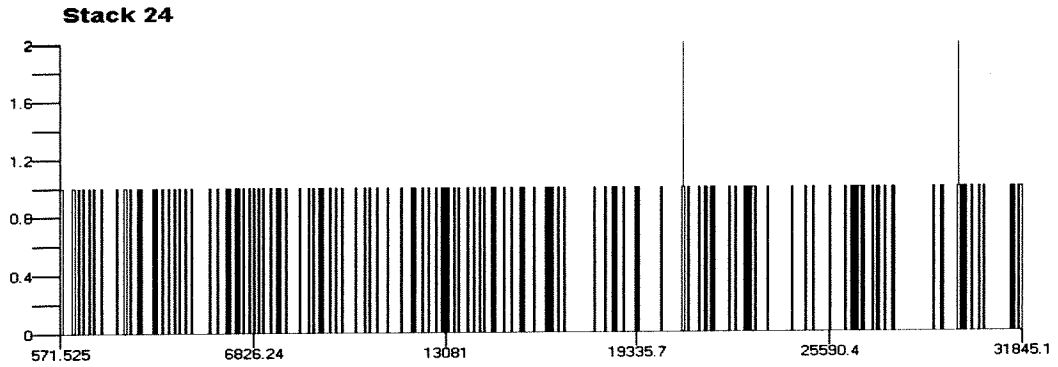


Figure 13.6, rgv queue on warehouse lane 24

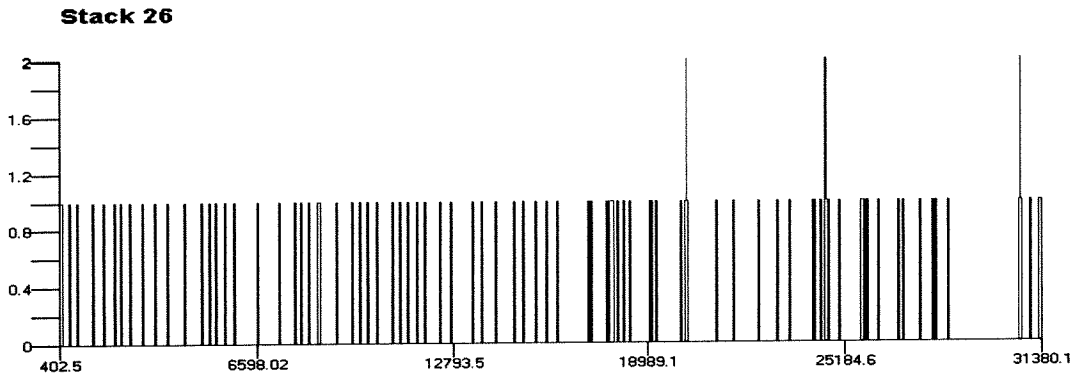


Figure 13.7, rgv queue on warehouse lane 26

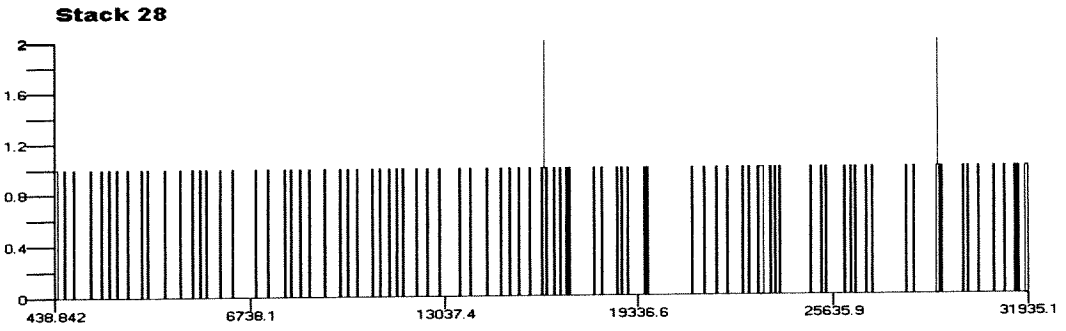


Figure 13.8, rgv queue on warehouse lane 28

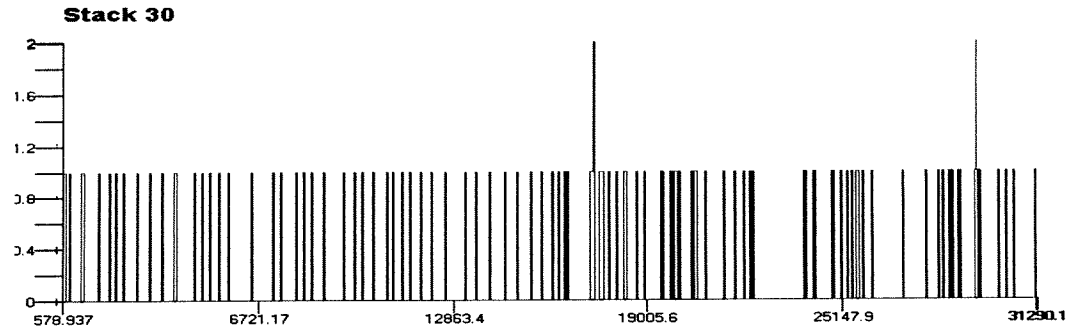


Figure 13.9 rgv queue on warehouse lane 30

Appendix 14: Input parameters for the experiment with a feeder at Berth 1, loading containers from an adjacent berth

The basic parameters used for the experiment with the feeder at Berth 1 when it is loaded with containers from the adjacent berth are shown in Table 14.1.
The parameter frombtob[2,1] indicates the number of containers in Yard 2 which can be used to load the feeder at Berth 1. Inplacecounter 1 indicates the number of containers in Yard 1 that can be used to load the feeder at Berth 1. In this experiment the inplacecounter 1 is 0, indicating the vessel at berth 1 will have to be loaded with containers from Yard 2. In this yard more than enough containers destined for berth 2 have been placed. The feeder unloads only 40 containers as can be seen by the import generator.
The parameters after the run are shown in Table 14.2.

Parameter	Value	
frombtob[1,2]	0	containers
frombtob[3,2]	0	containers
frombtob[2,1]	1600	containers
frombtob[2,3]	0	containers
inplacecounter 1	0	containers
inplacecounter 2	0	containers
inplacecounter 3	0	containers
seed of uniform	67	-
seed of arrival times feeders	17	-
seed of arrival times big	27	-
import generator feeders	40	containers
export generator feeders	1050	containers
import generator big vessels	0	containers
export generator big vessels	0	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
rgv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	70	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	58	seconds

Table 14.1: Input parameters for experiment: Feeder at Berth 1 loading from an adjacent berth

Parameter	Value	
frombtob[1,2]	40	containers
frombtob[2,1]	550	containers

Table 14.2, output after experiment: Feeder vessel at Berth 1 loading from an adjacent berth

In Table 14.2, frombtob[1,2] refers to the imported containers. After the run they are located in Yard 1 and destined to be taken away by a vessel from Berth 2. Frombtob[2,1] refers to the containers left in Yard 2 destined to be taken away by a vessel at Berth 1.

14.1.1 Crane occupancy rates

A feeder vessel moored at Berth 1 will have its containers adjacent loaded from Yard 2. The number of to be loaded containers is 1050. Considering that the number of containers that have been retrieved from the 20' stacks and the 40' stacks are the same means that 525 containers will be delivered to both stack types.

- Required 20' warehouse crane operational time: 525 * 80 seconds = 42,000 seconds
- Required 40' warehouse crane operational time: 525 * 80 seconds = 42,000 seconds

Number of warehouse cranes in Yard 2:
- Number of warehouse cranes in the 20' stacks in Yard 2 = 6 * 2 = 12 warehouse cranes
- Number of warehouse cranes in the 40' stacks in Yard 2 = 6 * 3 = 18 warehouse cranes

Needed operational time per warehouse crane:

- $42{,}000/12 = 3{,}500$ seconds per 20' stack warehouse crane
- $42{,}000/18 = 2{,}333$ seconds per 40' stack warehouse crane

Required unloading time for the feeder vessel: 5.1 hours ⇒ 18,360 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(3{,}500/18{,}360\right) * 100\% = 19.1\%$

40' stack: $\left(2{,}333/18{,}360\right) * 100\% = 12.7\%$

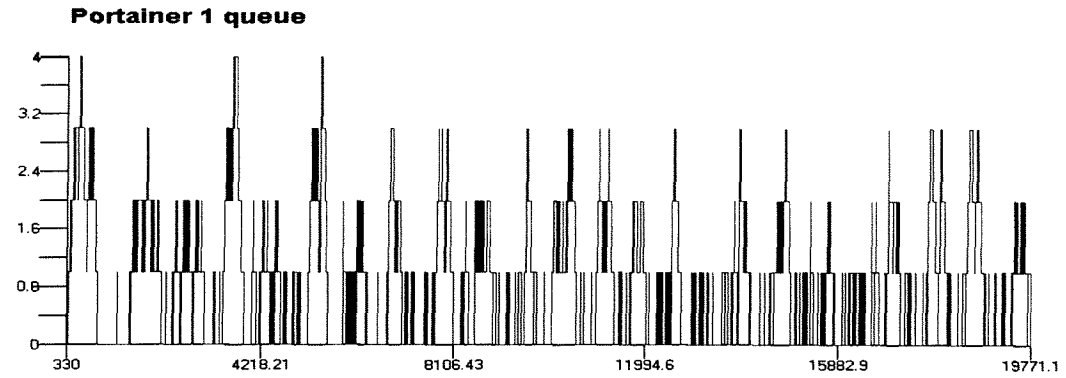


Figure 14.1, portainer 1 queue when loading a feeder from an adjacent Berth

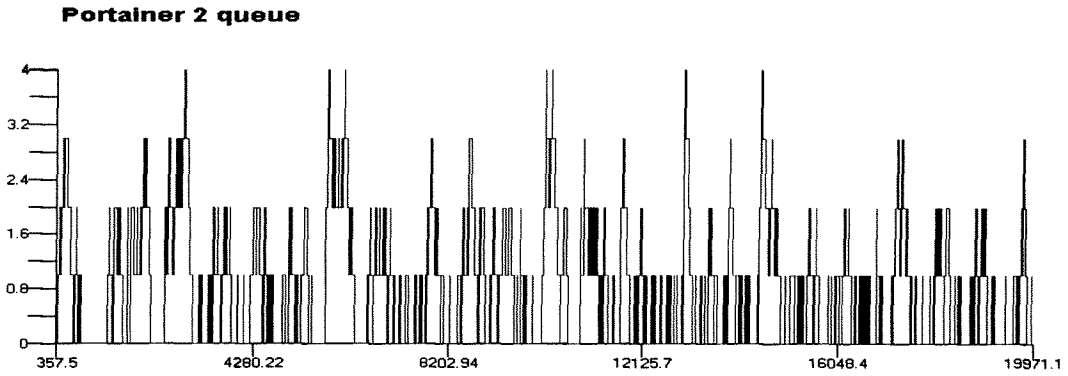


Figure 14.2, portainer 2 queue when loading a feeder from an adjacent Berth

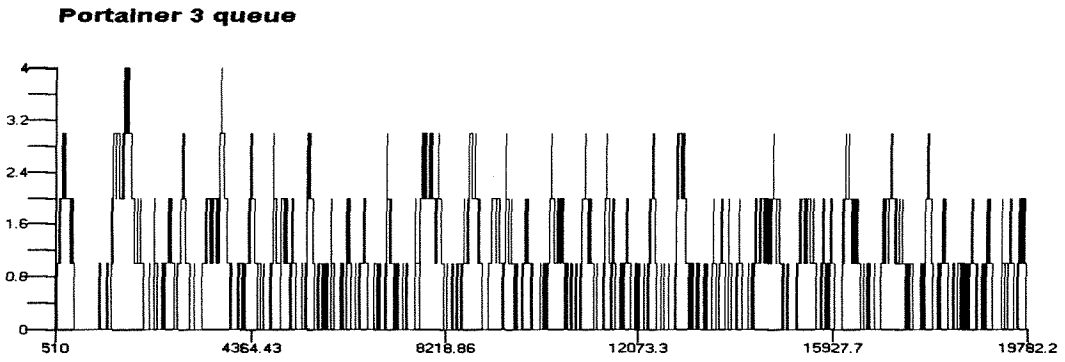


Figure14.3, portainer 3 queue when loading a feeder from an adjacent Berth

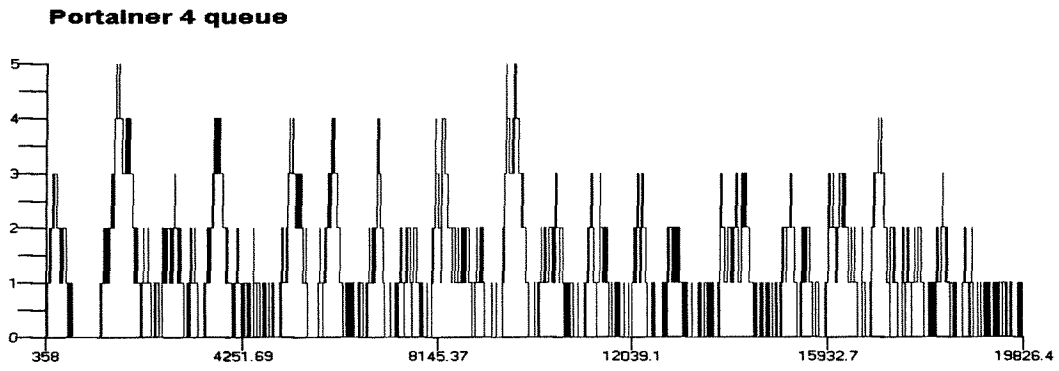


Figure 14.4, portainer 4 queue when loading a feeder from an adjacent Berth

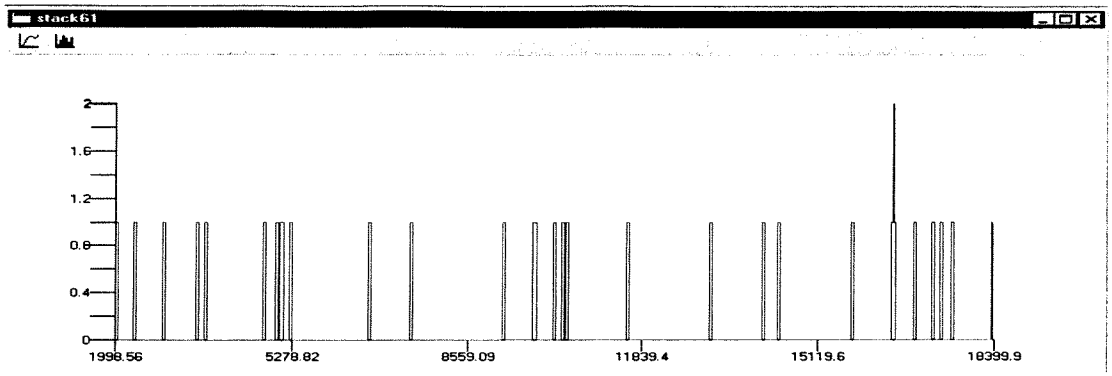


Figure 14.5, stack queue on warehouse lane 61 when adjacent loading a feeder at Berth 1

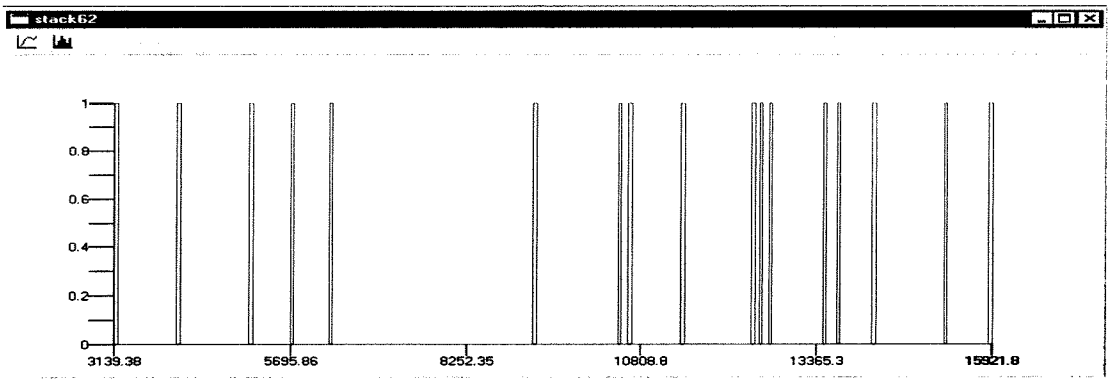
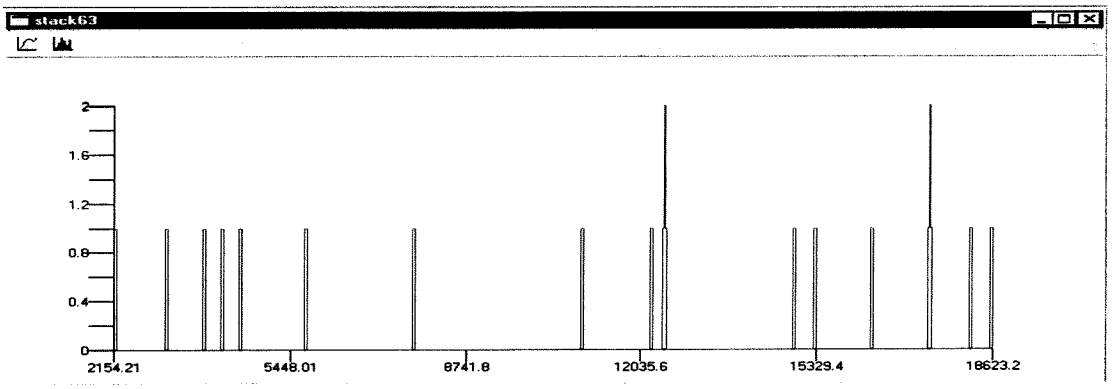


Figure 14.6, stack queue on warehouse lane 62 when adjacent loading a feeder at Berth 1



Ffigure 14.7, stack queue on warehouse lane 63 when adjacent loading a feeder at Berth 1

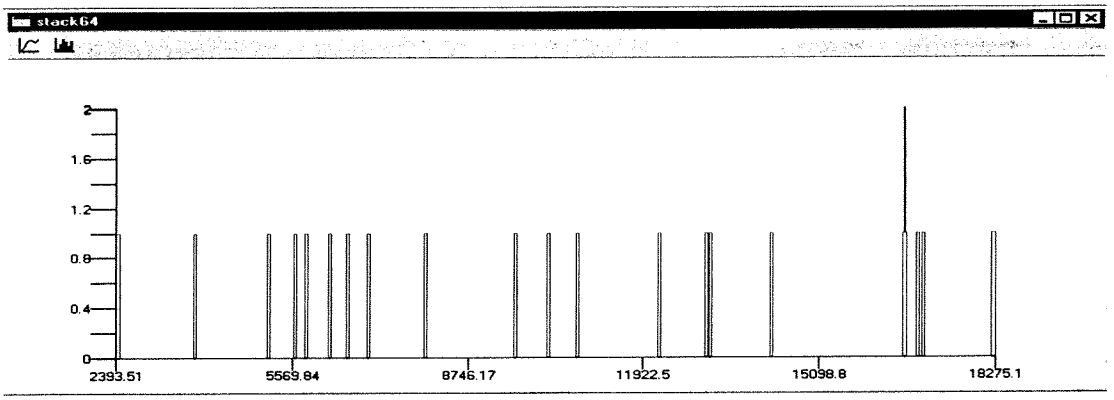


Figure 14.8, stack queue on warehouse lane 64 when adjacent loading a feeder at Berth 1

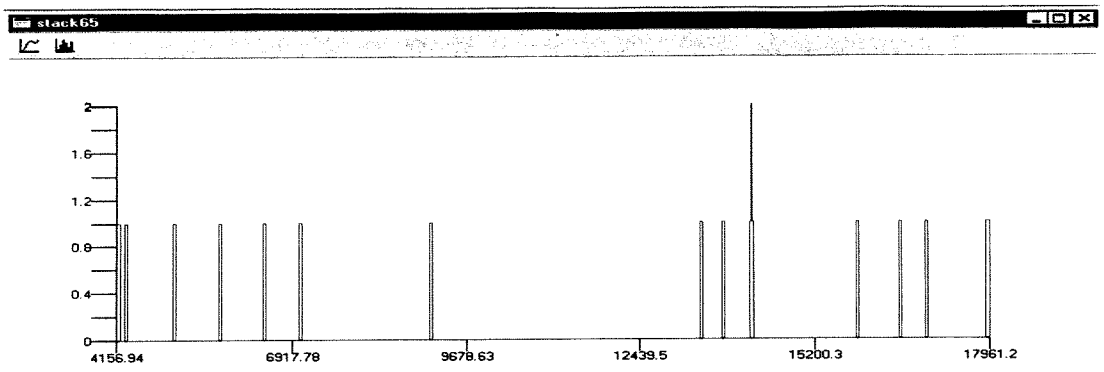


Figure 14.9, stack queue on warehouse lane 65 when adjacent loading a feeder at Berth 1

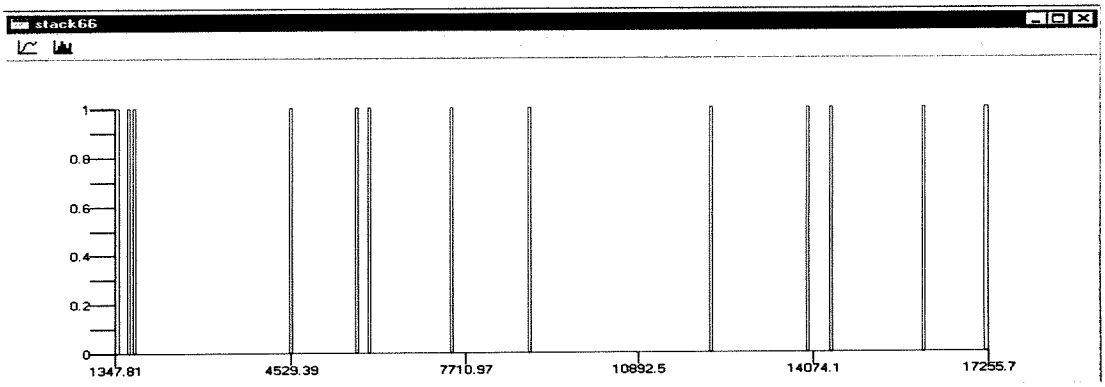


Figure 14.10, stack queue on warehouse lane 66 when adjacent loading a feeder at Berth 1

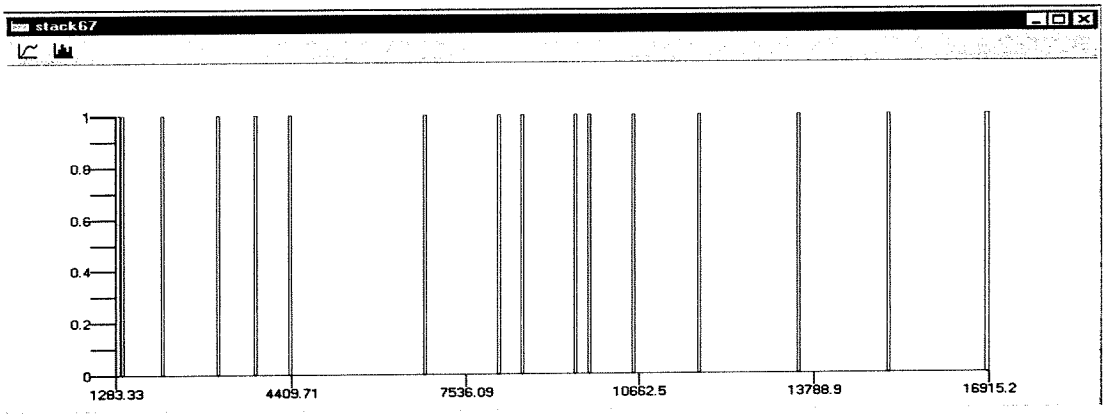


Figure 14.11, stack queue on warehouse lane 67 when adjacent loading a feeder at Berth 1

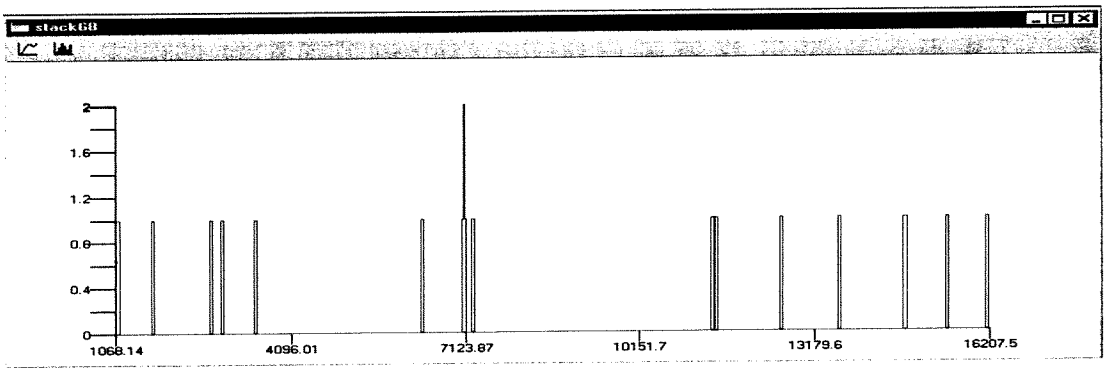


Figure 14.12, stack queue on warehouse lane 68 when adjacent loading a feeder at Berth 1

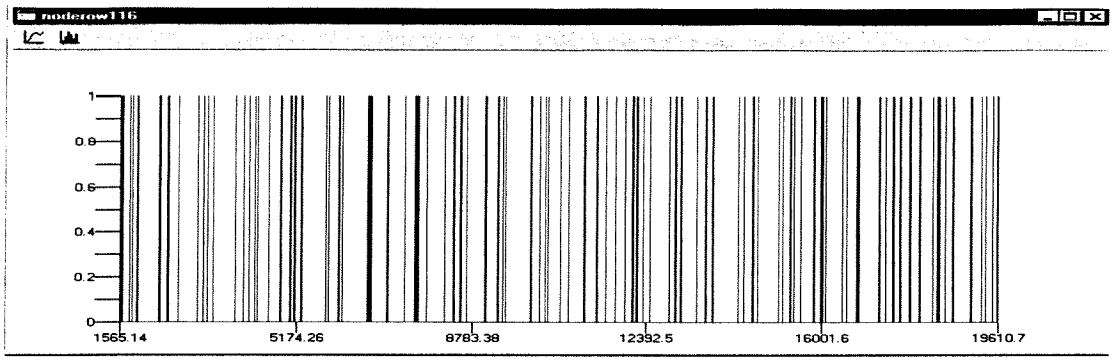


Figure 14.13, the rgv queues at node 116 when adjacent loading a feeder at Berth 1

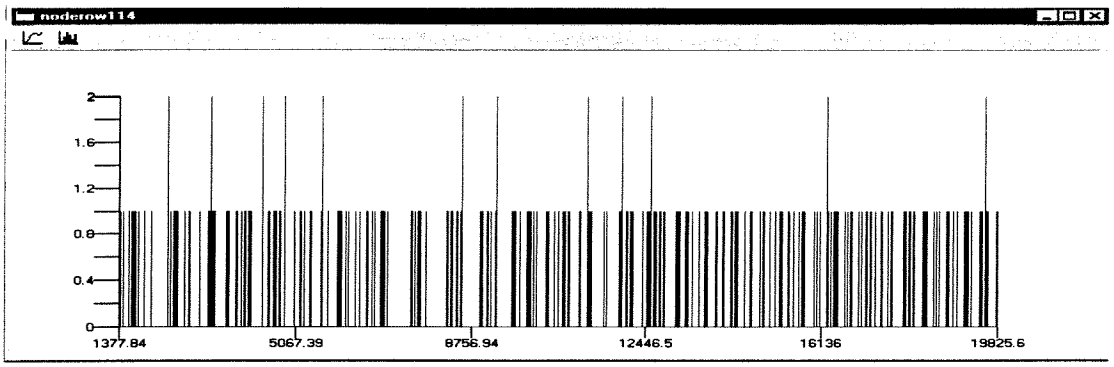


Figure 14.14, the rgv queues at node 114 when adjacent loading a feeder at Berth 1

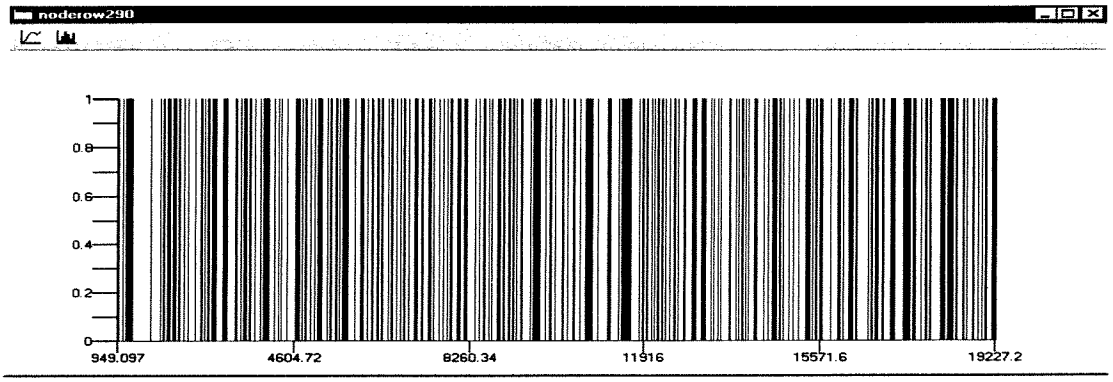


Figure 14.15, the grv queues at node 290 when adjacent loading a feeder at Berth 1

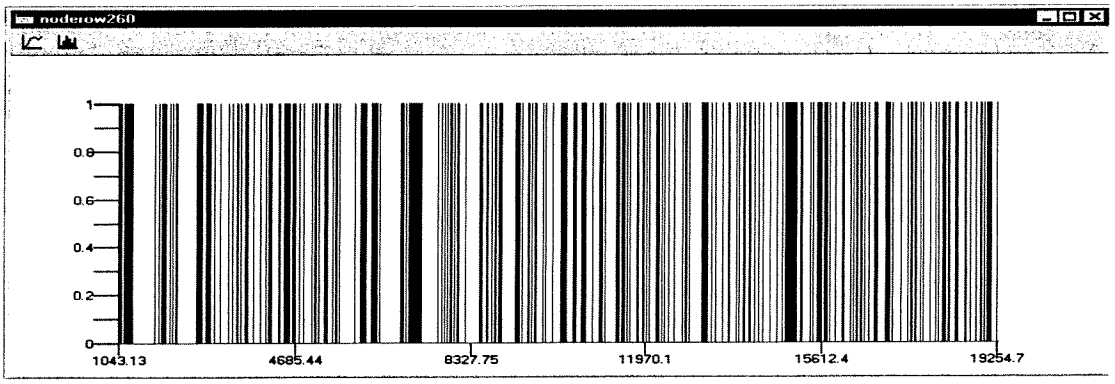


Figure 14.16, the rgv queues at node 260 when adjacent loading a feeder at Berth 1

Appendix 15: Input parameters for the experiment "shuffling from Berth 1"

The basic parameters used for this experiment are shown in Table 15.1. The shuffle process stops when there are no more containers in Yard 1 to be taken to Yard 2. The vessel has only 40 import containers and only 40 export containers. The parameters after the run are shown in Table 15.2.

Parameter	Value	
frombtob[1,2]	1200	containers
frombtob[3,2]	0	containers
frombtob[2,1]	1600	containers
frombtob[2,3]	0	containers
inplacecounter 1	50	containers
inplacecounter 2	0	containers
inplacecounter 3	0	containers
seed of uniform	67	-
seed of arrivaltimes feeders	17	-
seed of arrivaltimes big	27	-
import generator feeders	40	containers
export generator feeders	40	containers
import generator big vessels	0	containers
export generator big vessels	0	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
agv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	70	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	58	seconds

Table 15.1 Input parameters for experiment, shuffling from Berth 1

Parameter	Value	
frombtob[1,2]	0	containers
frombtob[3,2]	0	containers
frombtob[2,1]	350	containers
frombtob[2,3]	0	containers
Inplacecounter 1	1250	containers
Inplacecounter 2	1240	containers

Table 15.2 Output parameters for experiment shuffling from Berth 1

In Table 15.2 the parameters inplacecounter 1 & inplacecounter 2 indicate 1250+1240 containers in total have been shuffled to the right location.

15.1.1 Warehouse cranes located in Yard 1 while shuffling from Yard 1

The cranes located in Yard 1 will have to digest 2490 containers. 1250 containers to be delivered to Yard 2 while 1240 containers have to be delivered from Yard 2 to Yard 1.

Considering that the number of containers that will be retrieved and delivered form the 20' stack and the 40' stacks are the same means that 1245 containers will be handled by both stacks types.

- Required 20' warehouse crane operational time: 1245 * 80 seconds = 99,600 seconds
- Required 40' warehouse crane operational time: 1245 * 80 seconds = 99,600 seconds

Number of warehouse cranes in Yard 2:
- Number of warehouse cranes in the 20' stacks in Yard 1 = 4 * 2 = 8 warehouse cranes
- Number of warehouse cranes in the 40' stacks in Yard 2 = 4 * 3 = 12 warehouse cranes

Needed operational time per warehouse crane:

- $99{,}600/8 = 12{,}450$ seconds per 20' stack warehouse crane
- $99{,}600/12 = 8{,}300$ seconds per 40' stack warehouse crane

Required adjacent loading time for both feeder vessels: 7.4 hours $\Rightarrow$ 26,640 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(12{,}450/26{,}640\right) * 100\% = 46.7\ \%$

40' stack: $\left(8{,}300/26{,}640\right) * 100\% = 31.2\ \%$

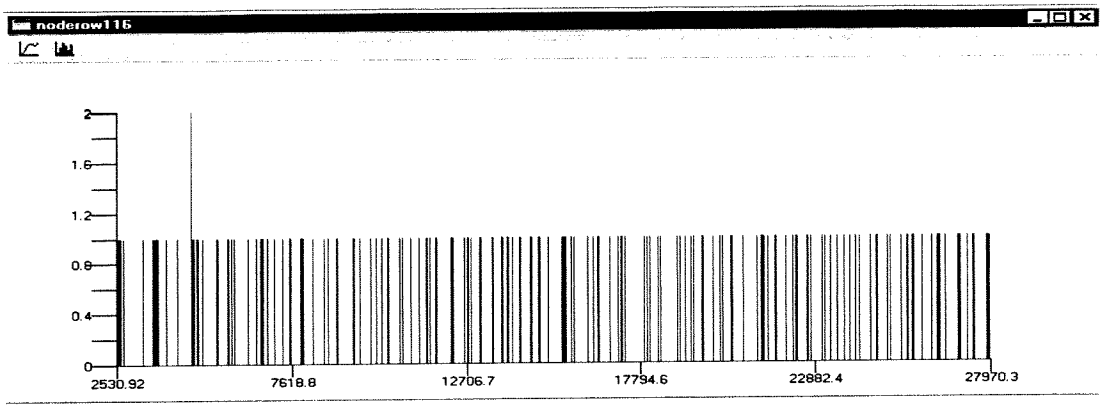


Figure 15.1, queue at node 116 when shuffling from Berth 1

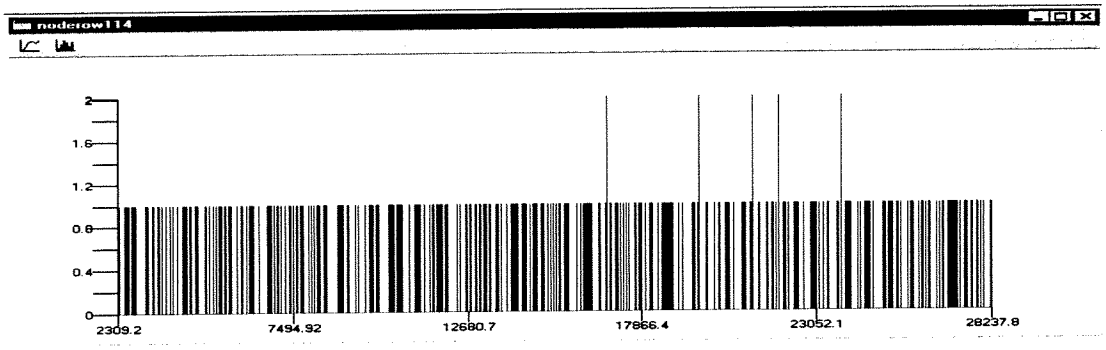


Figure 15.2, queue at node 114 when shuffling from Berth 1

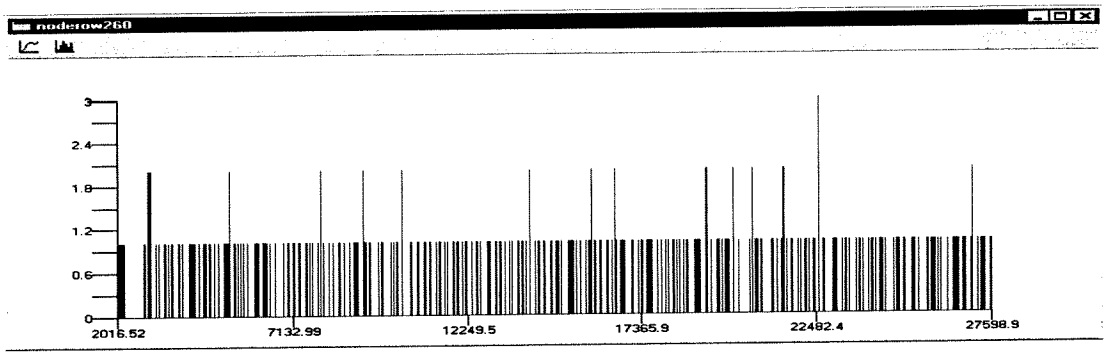


Figure 15.3, queue at node 260. when shuffling from Berth 1

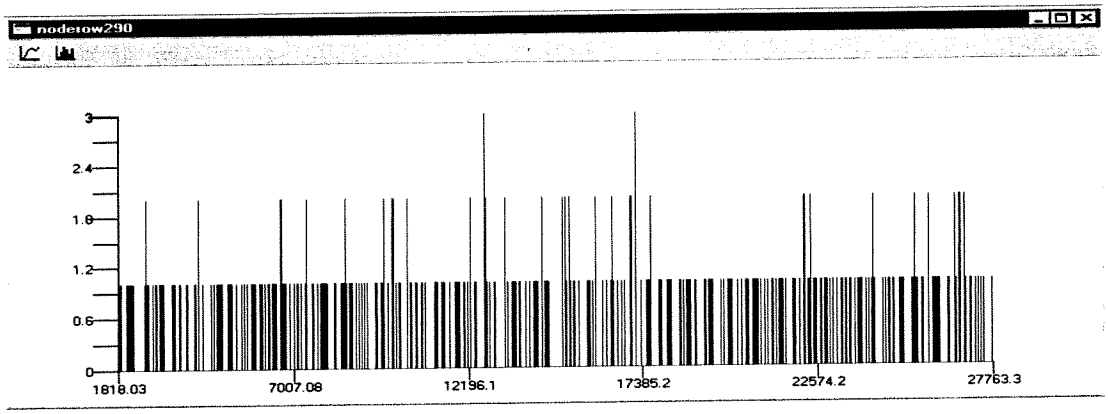


Figure 15.4, , queue at node 290 when shuffling from Berth 1

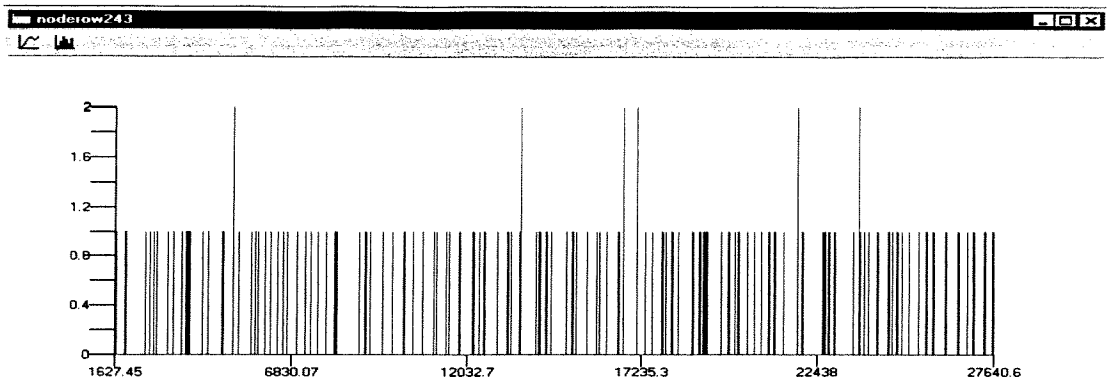


Figure 15.5, queue at node 243 when shuffling from Berth 1

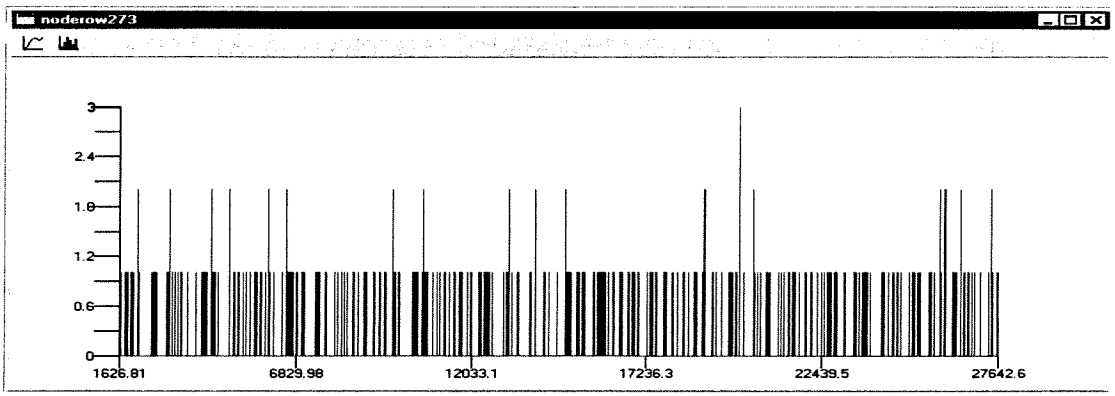


Figure 15.6, queue at node 273 when shuffling from Berth 1

Appendix 16: Input parameters for the experiment with one mega vessel at Berth 2 unloading

The basic parameters used for this experiment are shown in Table 16.1.
The changed parameters after the run are shown in Table 16.2.

Parameter	Value	
frombtob[1,2]	0	containers
frombtob[3,2]	0	containers
frombtob[2,1]	0	containers
frombtob[2,3]	0	containers
inplacecounter 1	0	containers
inplacecounter 2	0	containers
inplacecounter 3	0	containers
seed of uniform	67	-
seed of arrival times feeders	17	-
seed of arrival times big	27	-
import generator feeders	0	containers
export generator feeders	0	containers
import generator big vessels	3150	containers
export generator big vessels	3150	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
rgv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	70	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	58	seconds

Table 16.1: Input parameters for experiment: Mega vessel at Berth 2 unloading

Parameter	Value	
frombtob[2,1]	1579	containers
frombtob[2,3]	1571	containers

Table 16.2, output after experiment: Mega vessel at Berth 2 unloading

In Table 16.2 the parameters frombtob[2,1] and frombtob[2,3] indicate 1579 containers are unloaded eventually destined for Yard 1 and 1571 containers are unloaded destined for Yard 3.

16.1.1 Crane occupancy rates when unloading a mega vessel

A mega vessel moored at Berth 2 will have its containers unloaded in Yard 2. The number of unloaded containers is 3150. Considering that the number of containers that have been delivered to the 20' stacks and the 40' stacks are the same means that 1575 containers will be delivered to both stack types.

- Required 20' warehouse crane operational time: 1575 * 80 seconds = 126,000 seconds
- Required 40' warehouse crane operational time: 1575 * 80 seconds = 126,000 seconds

Number of warehouse cranes in Yard 2:
- Number of warehouse cranes in the 20' stacks in Yard 2 = 6 * 2 = 12 warehouse cranes
- Number of warehouse cranes in the 40' stacks in Yard 2 = 6 * 3 = 18 warehouse cranes

Needed operational time per warehouse crane:

- $126{,}000/12 = 10{,}500$ seconds per 20' stack warehouse crane
- $126{,}000/18 = 7{,}000$ seconds per 40' stack warehouse crane

Required unloading time for the mega vessel: 8.9 hours $\Rightarrow$ 32,040 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(10{,}500/32{,}040\right) * 100\% = 32.8\,\%$

40' stack: $\left(7{,}000/32{,}040\right) * 100\% = 21.8\,\%$

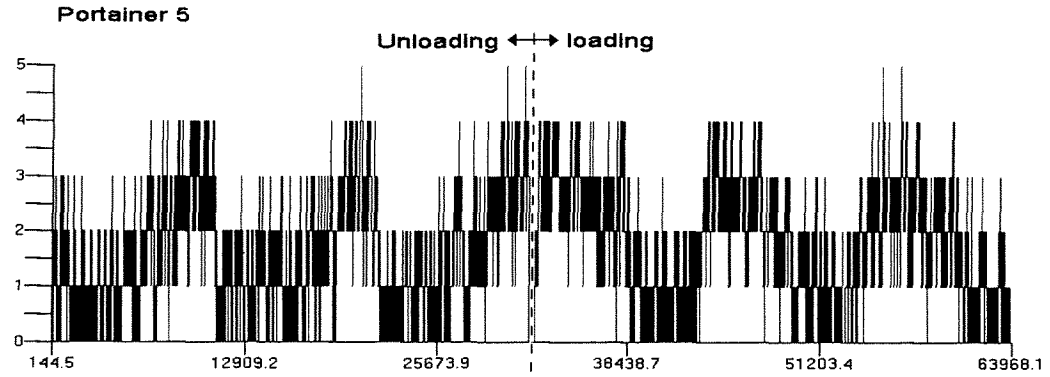


Figure 16.1, portainer 5 queue at Berth 2 when unloading and loading a mega vessel

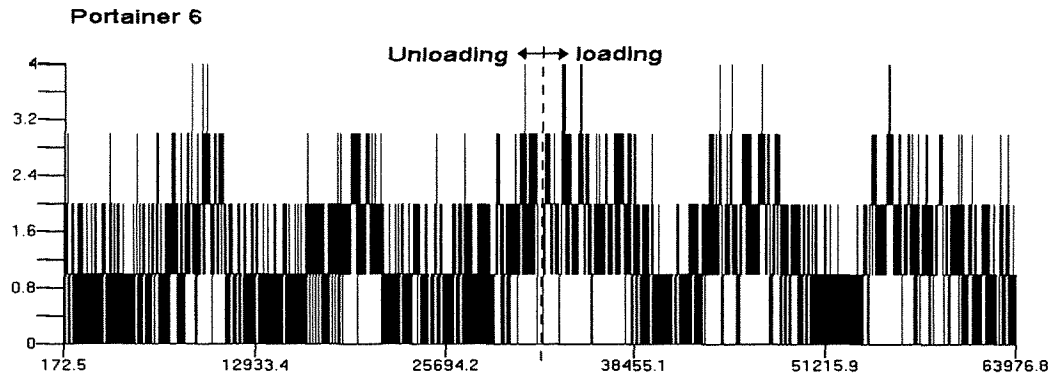


Figure 16.2, portainer 6 queue at Berth 2 when unloading and loading a mega vessel

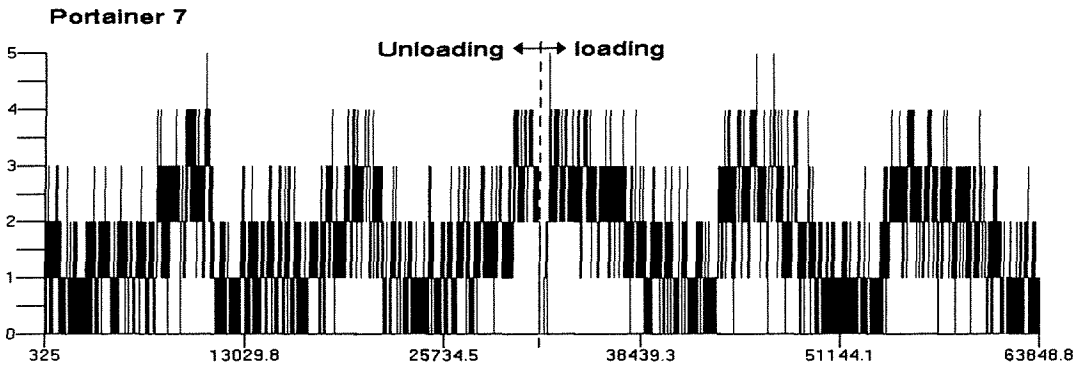


Figure 16.3, portainer 7 queue at Berth 2 when unloading and loading a mega vessel

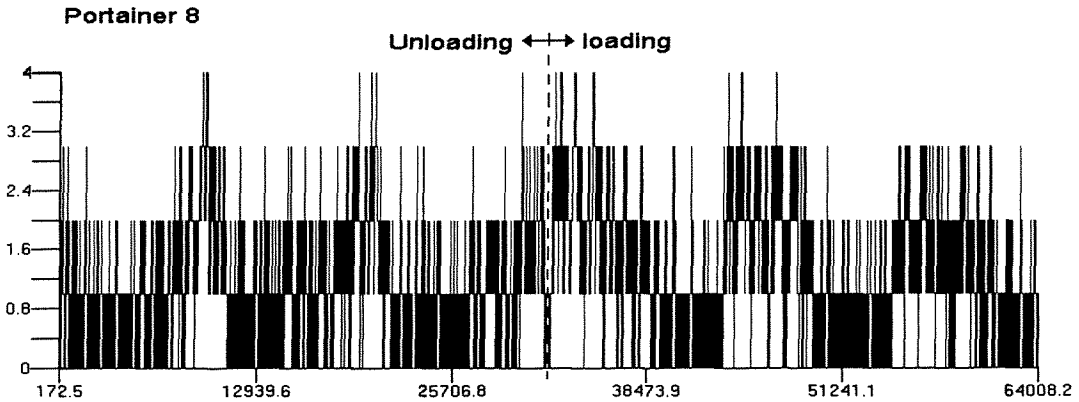


Figure 16.4, portainer 8 queue at Berth 2 when unloading and loading a mega vessel

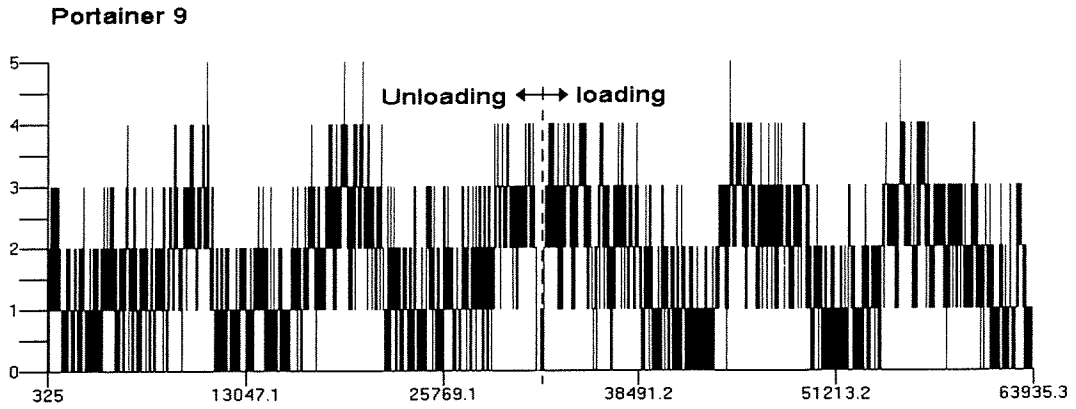


Figure 16.5, portainer 9 queue at Berth 2 when unloading and loading a mega vessel

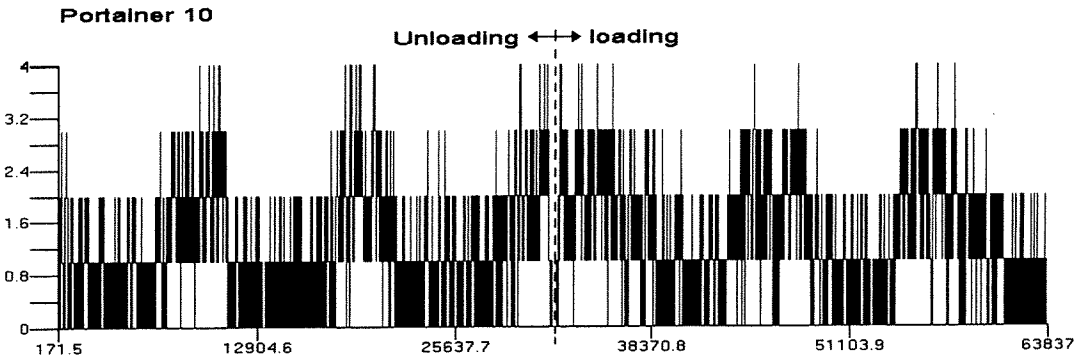


Figure 16.6, portainer 10 queue at Berth 2 when unloading and loading a mega vessel

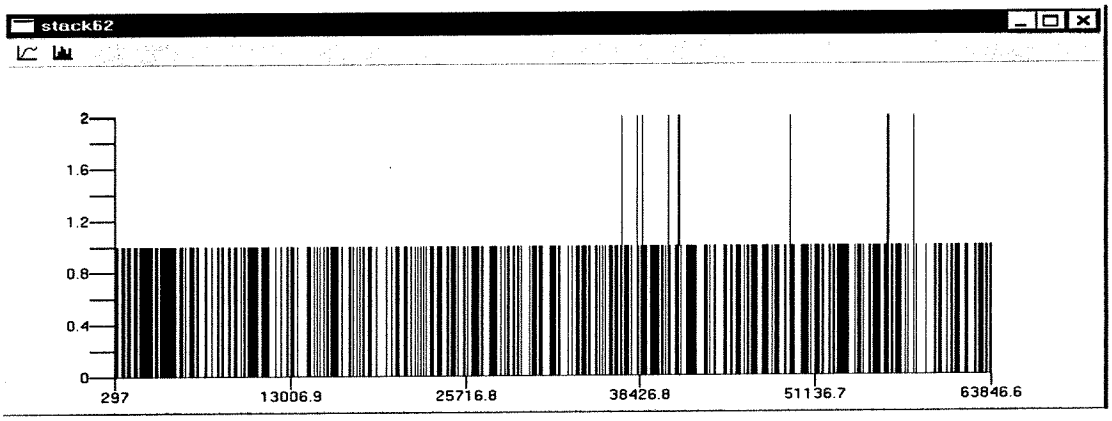


Figure 16.7, rgv queues on warehouse lane 62 when loading and unloading a mega vessel

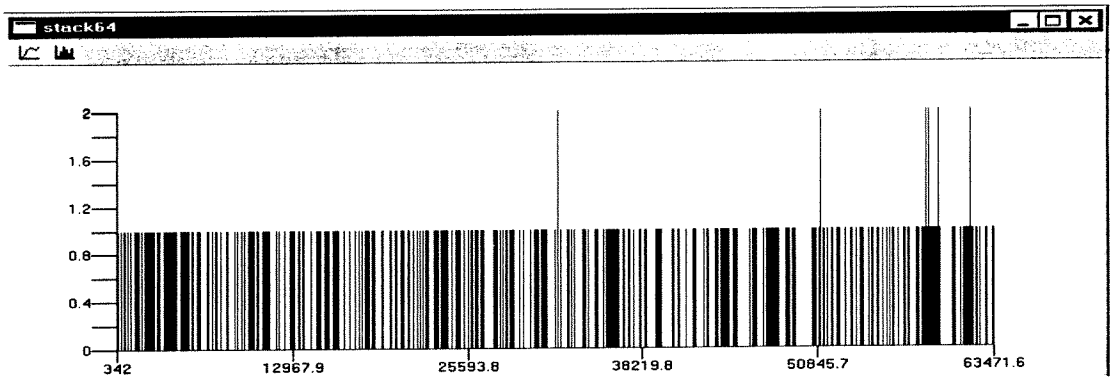


Figure 16.8, rgv queues on warehouse lane 64 when loading and unloading a mega vessel

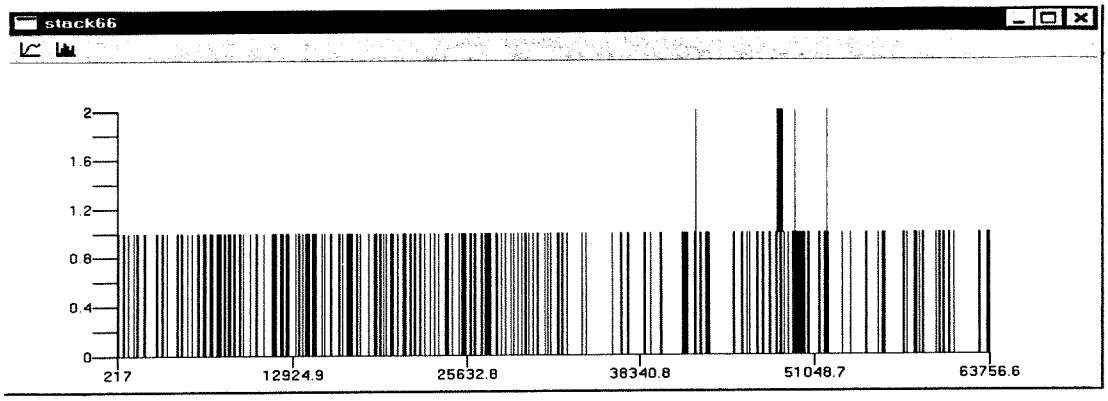


Figure 16.9, rgv queues on warehouse lane 66 when loading and unloading a mega vessel

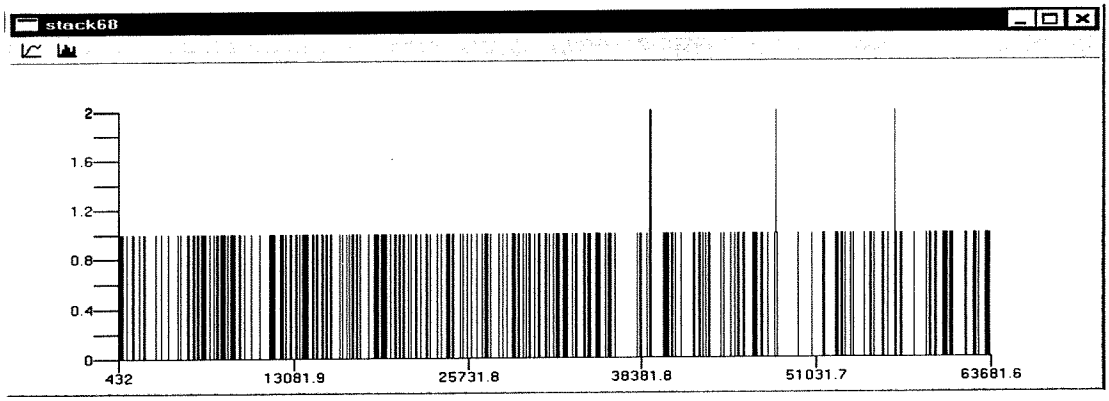


Figure 16.10, rgv queues on warehouse lane 68 when loading and unloading a mega vessel

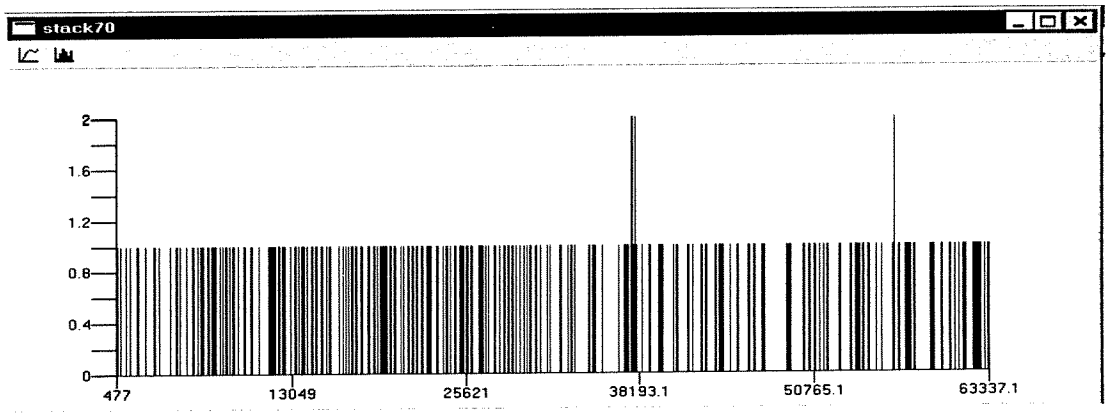


Figure 16.11, rgv queues on warehouse lane 70 when loading and unloading a mega vessel

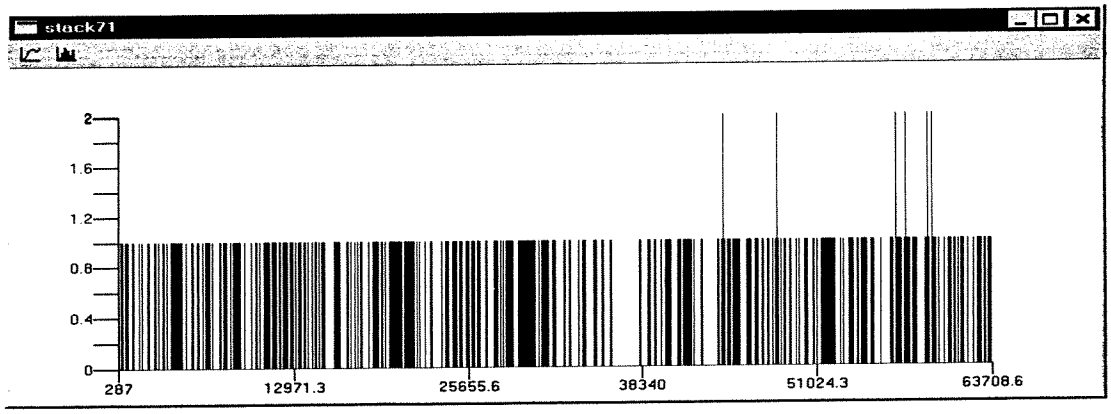


Figure 16.12, rgv queues on warehouse lane 71 when loading and unloading a mega vessel

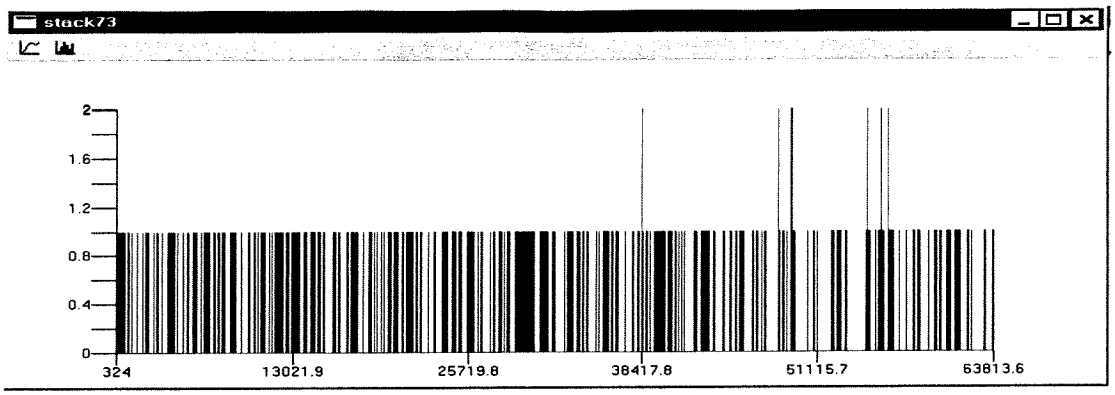


Figure 16.13, rgv queues on warehouse lane 73 when loading and unloading a mega vessel

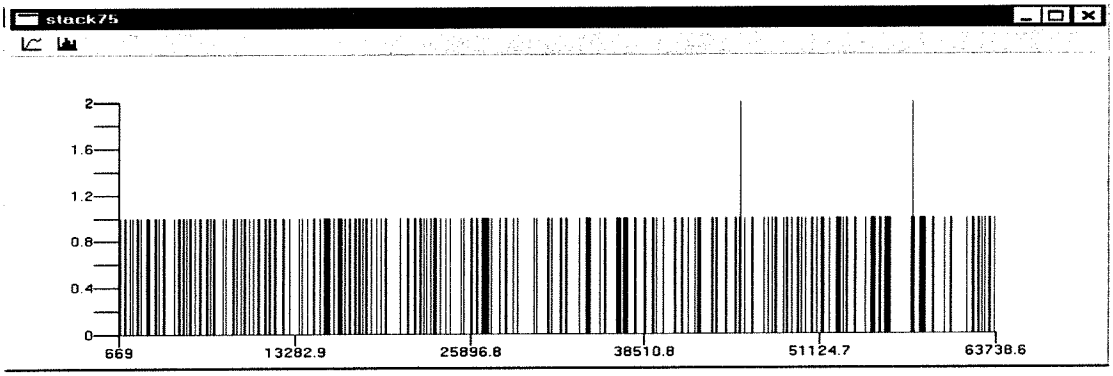


Figure 16.14, rgv queues on warehouse lane 75 when loading and unloading a mega vessel

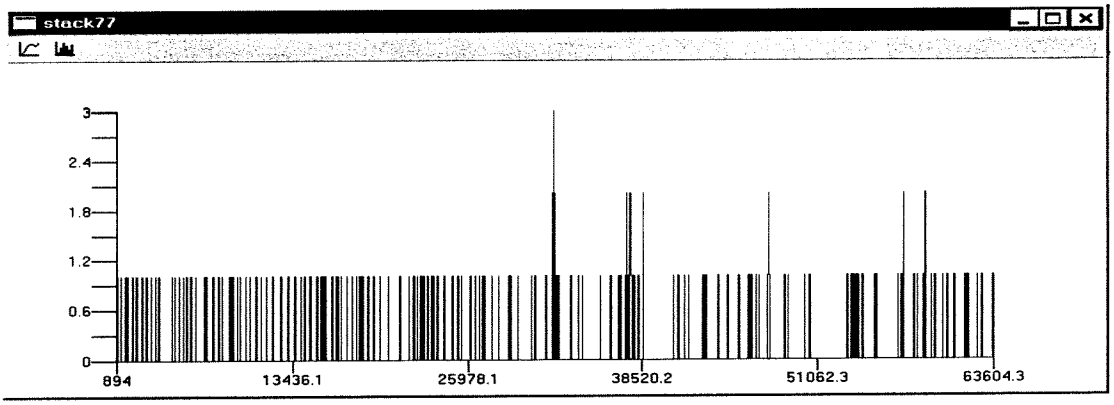


Figure 16.15, rgv queues on warehouse lane 77 when loading and unloading a mega vessel

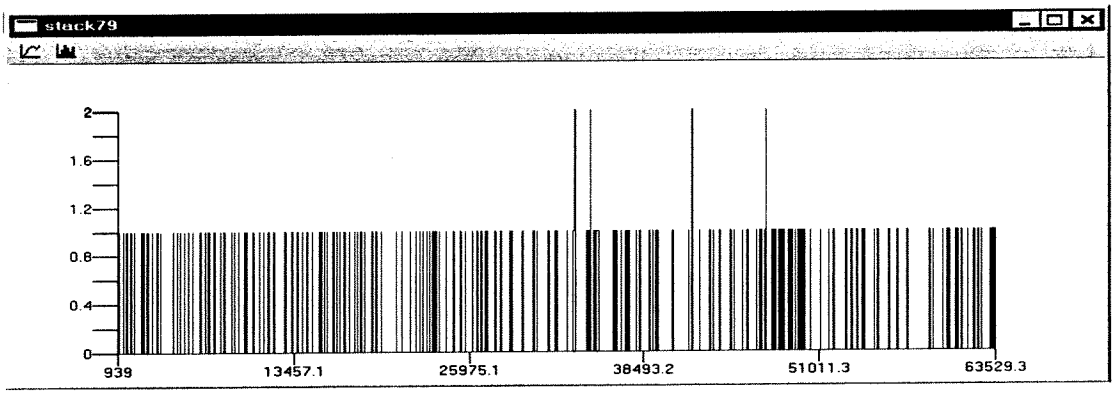


Figure 16.16, rgv queues on warehouse lane 79 when loading and unloading a mega vessel

Appendix 17: Input parameters for the experiment with one mega vessel at Berth 2, loading

The basic parameters used for the experiment with the mega vessel at Berth 2 when it is loading are shown in Table 17.1.
The parameters after the run are shown in Table 17.2.
Inplacecounter 2 refers to the number of containers in yard 2 destined to be taken away by a vessel at berth 2. These containers can be loaded fast. In this experiment there are 10000 containers ready in yard 2 to be taken away.

Parameter	Value	
frombtob[1,2]	0	containers
frombtob[3,2]	0	containers
frombtob[2,1]	0	containers
frombtob[2,3]	0	containers
inplacecounter 1	0	containers
inplacecounter 2	10000	containers
inplacecounter 3	0	containers
seed of uniform	67	-
seed of arrivaltimes feeders	17	-
seed of arrivaltimes big	27	-
import generator feeders	0	containers
export generator feeders	0	containers
import generator big vessels	3150	containers
export generator big vessels	3150	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
rgv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	70	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	58	seconds

Table 17.1: Input parameters for experiment: Mega vessel at berth 2 loading

Parameter	Value	
frombtob[2,1]	1571	containers
frombtob[2,3]	1579	containers
Inplacecounter 2	6850	containers

Table 17.2, output after experiment: Mega vessel at berth 2 loading

In Table 17.2 the parameters frombtob[2,1] and frombtob[2,3] indicate 1579 containers are unloaded desitined for Berth 1 and 1571 containers are unloaded destined for Yard 3.
Inplacecounter 2 is 6850 this implies 3150 containers have been taken away by the vessel.

17.1.1 Crane occupancy rates when loading a mega vessel

A mega vessel moored at Berth 2 will have its containers fast loaded from Yard 2. The number of to be loaded containers is 3150. Considering that the number of containers that have have retrieved from the 20' stacks and the 40' stacks are the same means that 1575 containers will be delivered to both stack types.

- Required 20' warehouse crane operational time: 1575 * 80 seconds = 126,000 seconds
- Required 40' warehouse crane operational time: 1575 * 80 seconds = 126,000 seconds

Number of warehouse cranes in Yard 2:
- Number of warehouse cranes in the 20' stacks in Yard 2 = 6 * 2 = 12 warehouse cranes
- Number of warehouse cranes in the 40' stacks in Yard 2 = 6 * 3 = 18 warehouse cranes

Needed operational time per warehouse crane:

- $126{,}000/12 = 10{,}500$ seconds per 20' stack warehouse crane
- $126{,}000/18 = 7{,}000$ seconds per 40' stack warehouse crane

Required unloading time for the mega vessel: 8.9 hours $\Rightarrow$ 32,040 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(10{,}500/32{,}040\right) * 100\% = 32.8\,\%$

40' stack: $\left(7{,}000/32{,}040\right) * 100\% = 21.8\,\%$

Appendix 18: Input parameters for the experiment with one mega vessel at Berth 2 unloading and loading containers from an adjacent berth

The basic parameters used for the experiment with the mega vessel at Berth 2 when it is being loaded from two adjacent berths are shown in Table 18.1.
The parameter frombtob[1,2] indicates the number of containers in Yard 1 which can be used to load the vessel at Berth 2. The same principle applies for frombtob[3,2], frombtob[2,1] and frombtob[2,3]. Inplacecounter 1 indicates the number of containers in Yard 1 that can be used to load a vessel at Berth 1. In this experiment Inplacecounter 2 is 0, indicating the vessel at Berth 2 will have to be loaded with containers from Yard 1 and Yard 3. In these yards more than enough containers destined for berth 2 have been placed.
The changed parameters after the run are shown in Table 18.2.

Parameter	Value	
frombtob[1,2]	10000	containers
frombtob[3,2]	10000	containers
frombtob[2,1]	0	containers
frombtob[2,3]	0	containers
inplacecounter 1	0	containers
inplacecounter 2	0	containers
inplacecounter 3	0	containers
seed of uniform	67	-
seed of arrivaltimes feeders	17	-
seed of arrivaltimes big	27	-
import generator feeders	0	containers
export generator feeders	0	containers
import generator big vessels	3150	containers
export generator big vessels	3150	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
rgv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	70	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	58	seconds

Table 18.1: Input parameters for experiment: Mega vessel at Berth 2 loading from an adjacent berth

Parameter	Value	
frombtob[1,2]	8453	containers
frombtob[3,2]	8397	containers
frombtob[2,1]	1579	containers
frombtob[2,3]	1571	containers

Table 18.2, output after experiment: Mega vessel at Berth 2 loading from an adjacent berth

In Table 18.2frombtob[2,1] is 8453. This means that from the original 10000 containers in Yard 1 destined for a vessel at Berth 2, 8453 containers are left in Yard 1, indicating that: 10000-8453=1547 containers from Yard 1 are used to load the mega vessel at Berth 2.

18.1.1 Crane occupancy rates

A megavessel moored at Berth 2 will have its containers adjacent loaded from Yard 1 & 3. The number of to be loaded containers is 3150. Considering that the number of containers that have been retrieved from the 20' stacks and the 40' stacks are the same means that 1575 containers will be delivered to both stack types in Yard 1 & 3.

- Required 20' warehouse crane operational time: 1575 * 80 seconds = 126,000 seconds
- Required 40' warehouse crane operational time: 1575 * 80 seconds = 126,000 seconds

Number of warehouse cranes in Yard 1 & 3:
- Number of warehouse cranes in the 20' stacks = 8 * 2 = 16 warehouse cranes
- Number of warehouse cranes in the 40' stacks = 8 * 3 = 24 warehouse cranes

Needed operational time per warehouse crane:

- $126{,}000/16 = 7{,}875$ seconds per 20' stack warehouse crane
- $126{,}000/24 = 5{,}250$ seconds per 40' stack warehouse crane

Required unloading time for the feeder vessel: 9.7 hours $\Rightarrow$ 34,920 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(7{,}875/34{,}920\right) * 100\% = 22.6\,\%$

40' stack: $\left(5{,}250/34{,}920\right) * 100\% = 15.0\,\%$

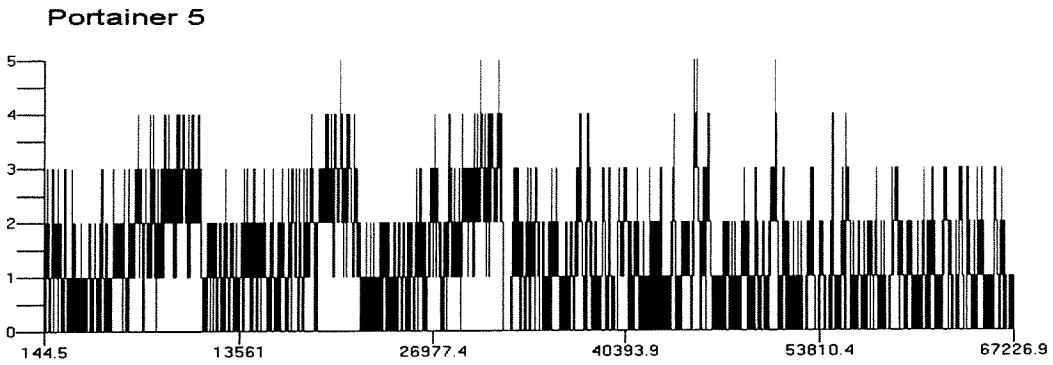


Figure 18.1, portainer 5 queue when unloading and loading from an adjacent berth at Berth 2

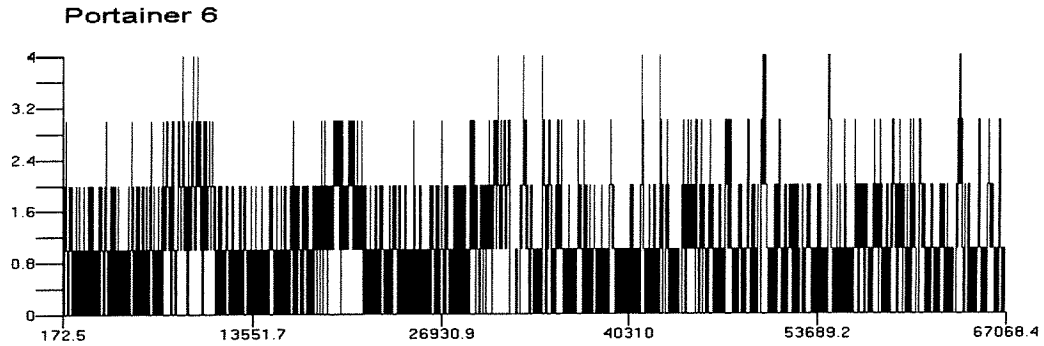


Figure 18.2, portainer 6 queue when unloading and loading from an adjacent berth at Berth 2

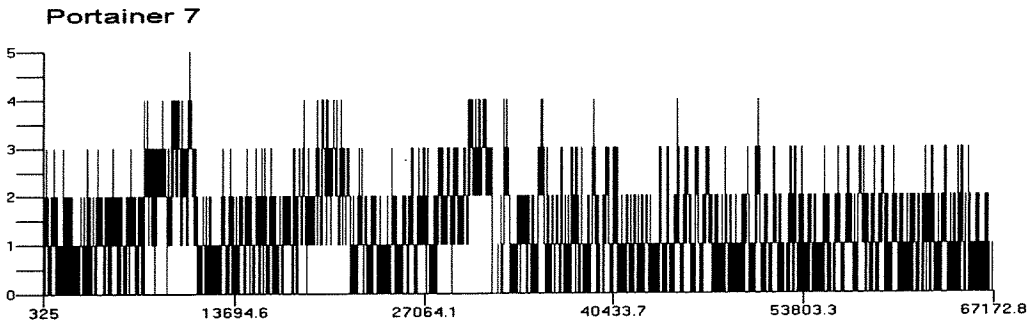


Figure 18.3, portainer 7 queue when unloading and loading from an adjacent berth at Berth 2

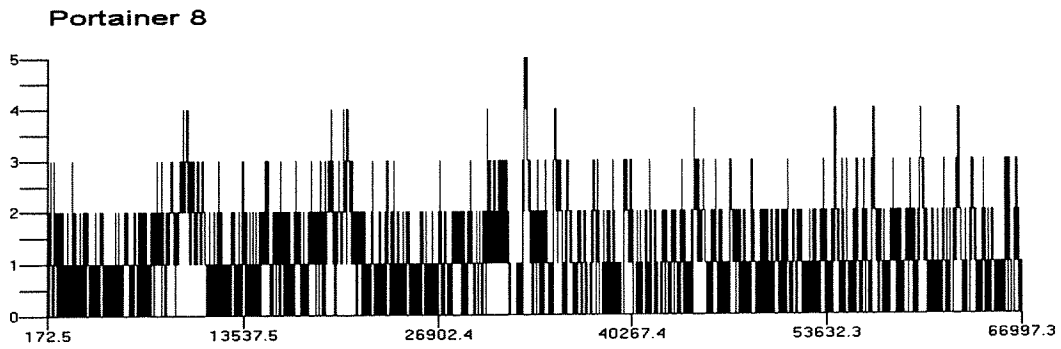


Figure 18.4, portainer 8 queue when unloading and loading from an adjacent berth at Berth 2

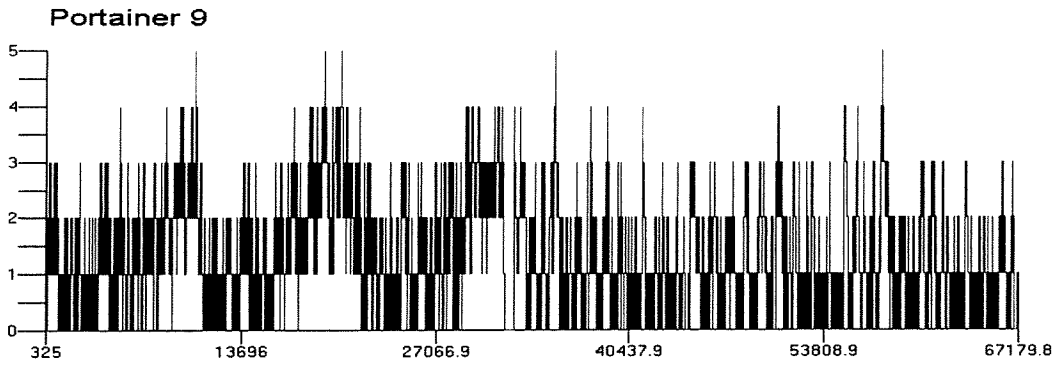


Figure 18.5, portainer 9 queue when unloading and loading from an adjacent berth at Berth 2

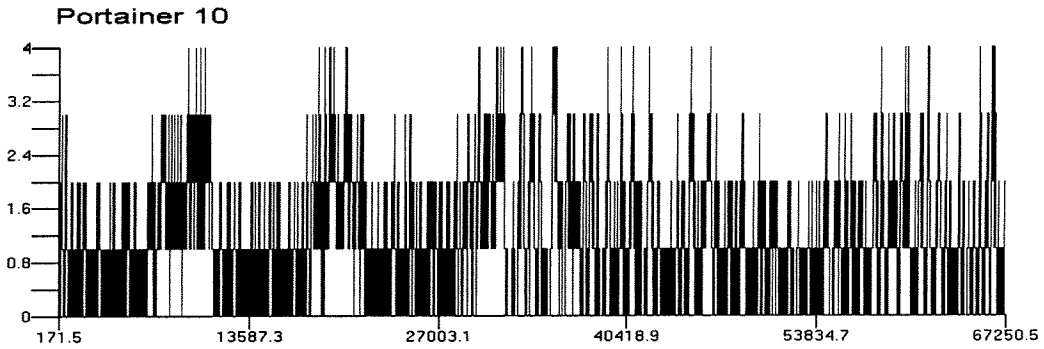


Figure 18.6, portainer 10 queue when unloading and loading from an adjacent berth at Berth 2

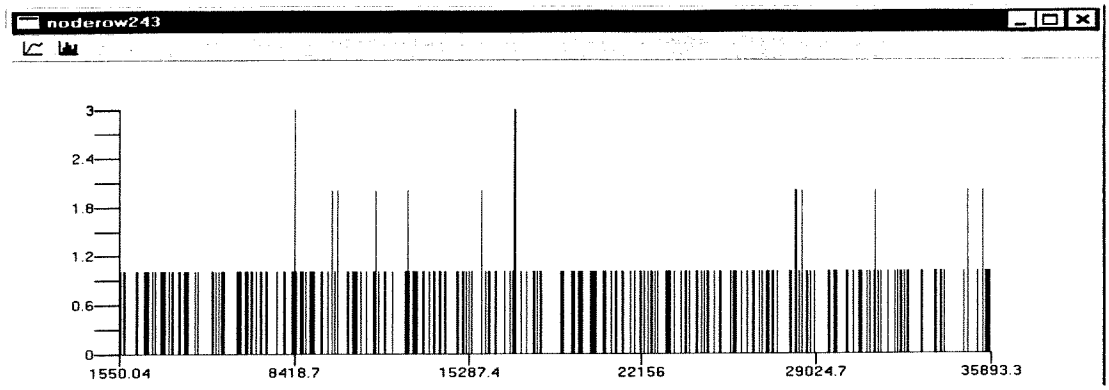


Figure 18.7, node row at node 243 when adjacent loading at Berth 2

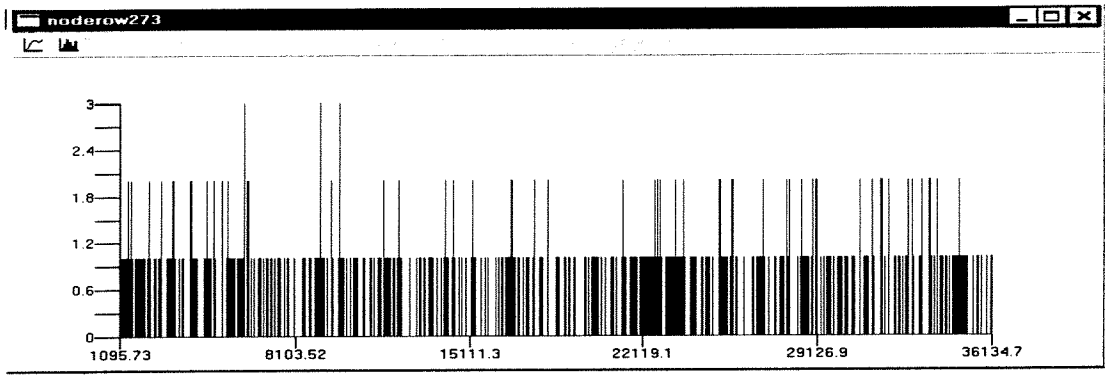


Figure 18.8, node row at node 273 when adjacent loading at Berth 2

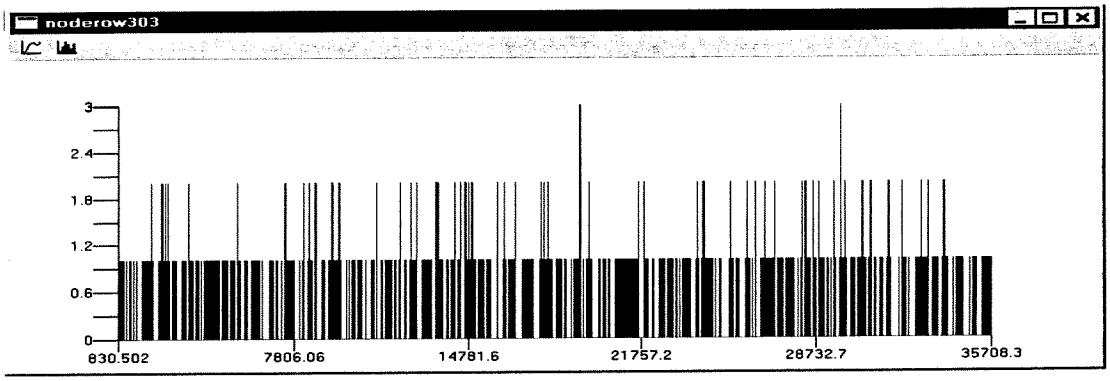


Figure 18.9, node row at node 303 when adjacent loading at Berth 2

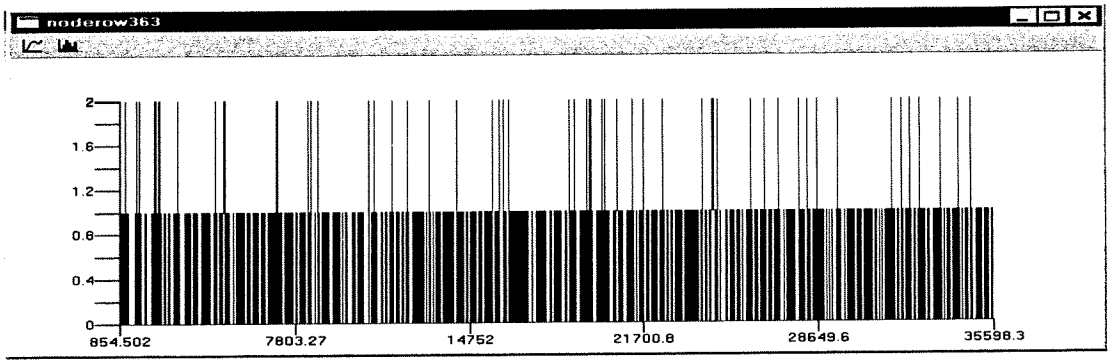


Figure 18.10, node row at node 363 when adjacent loading at Berth 2

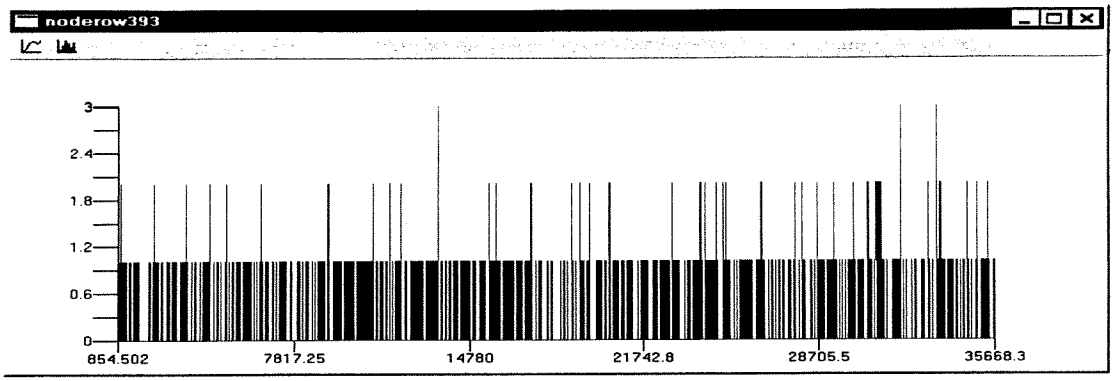


Figure 18.11, node row at node 393 when adjacent loading at Berth 2

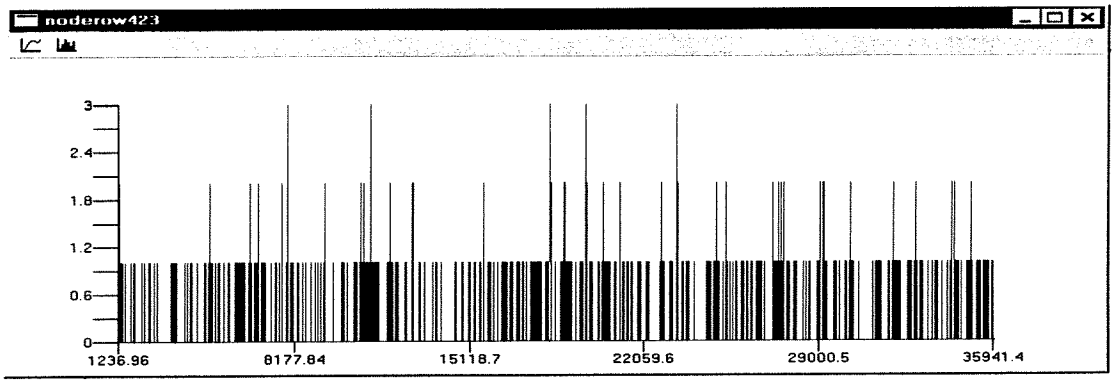


Figure 18.12, node row at node 423 when adjacent loading at Berth 2

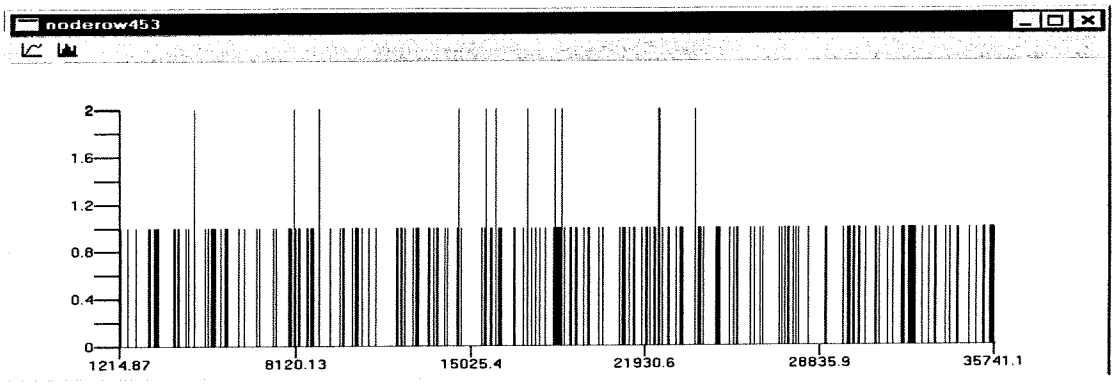


Figure 18.13, node row at node 453 when adjacent loading at Berth 2

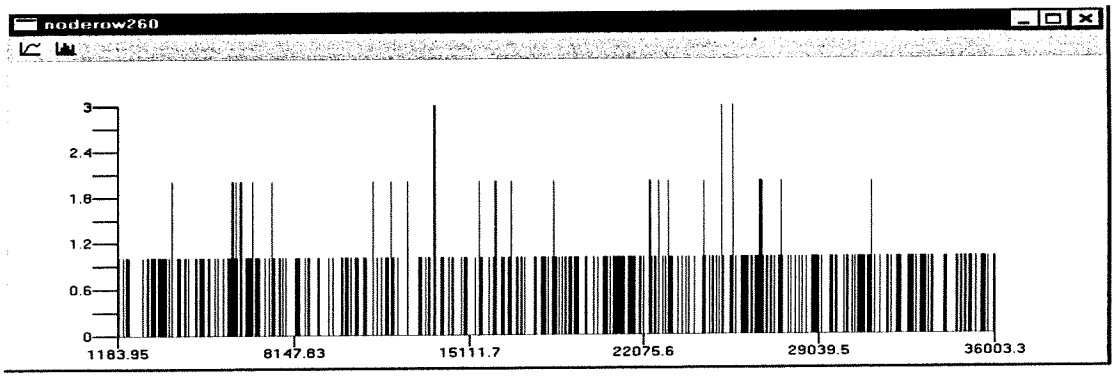


Figure 18.14, node row at node 260 when adjacent loading at Berth 2

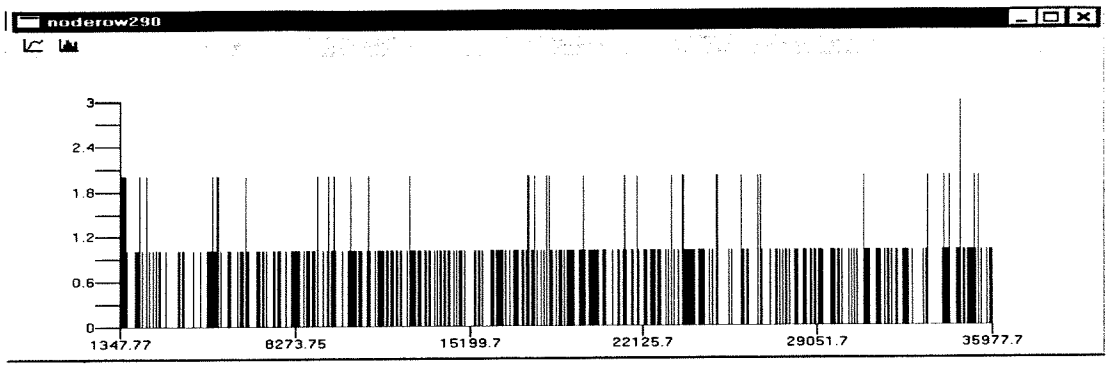


Figure 18.15, node row at node 290 when adjacent loading at Berth 2

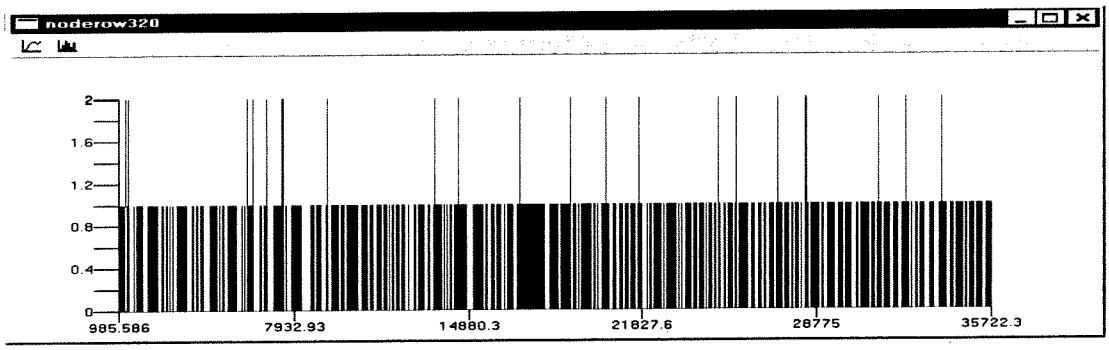


Figure 18.16, node row at node 320 when adjacent loading at Berth 2

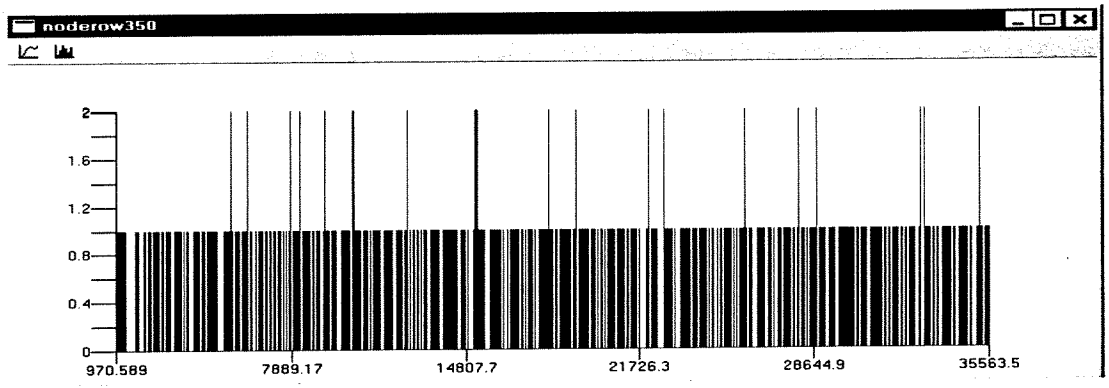


Figure 18.17, node row at node 350 when adjacent loading at Berth 2

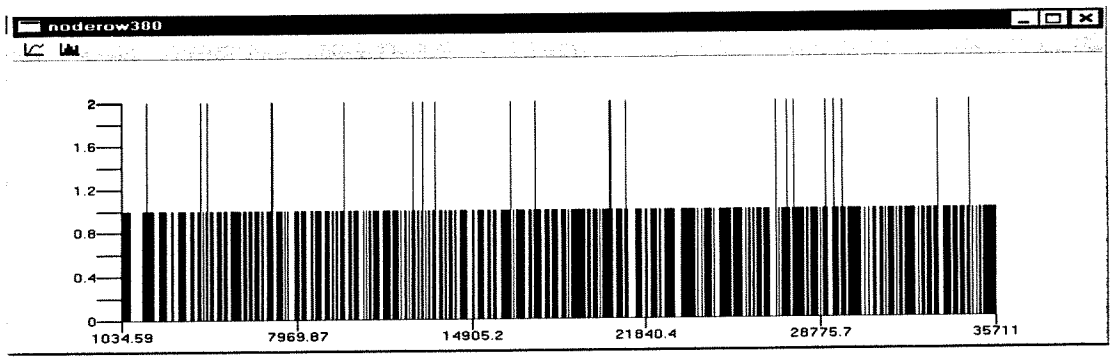


Figure 18.18, node row at node 380 when adjacent loading at Berth 2

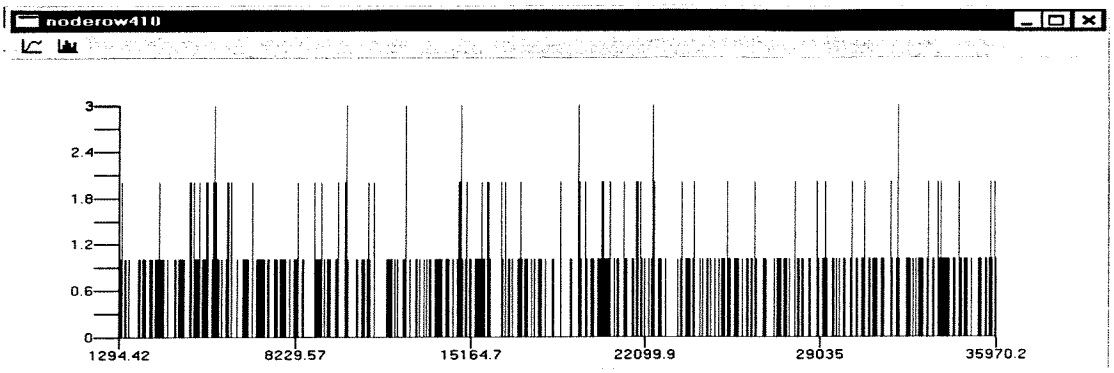


Figure 18.19, node row at node 410 when adjacent loading at Berth 2

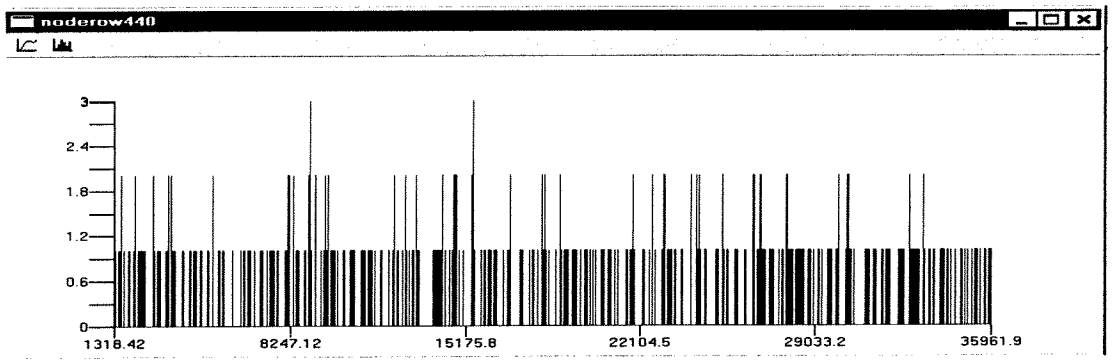


Figure 18.20, node row at node 440 when adjacent loading at Berth 2

Appendix 19: Input parameters for the experiment "shuffling from Berth 2"

The basic parameters used for the experiment shuffling from Berth 2 are shown in Table 19.1. The shuffle process stops when there are no more containers in Yard 2 to be taken to Yard 1 or 3. In this process there are always containers available in Yard 1 and Yard 3 while frombtob[1,2] and frombtob[3,2] are 10000 each.
The changed parameters after the run are shown in Table 19.2.

Parameter	Value	
frombtob[1,2]	10000	containers
frombtob[3,2]	10000	containers
frombtob[2,1]	1575	containers
frombtob[2,3]	1575	containers
inplacecounter 1	0	containers
inplacecounter 2	0	containers
inplacecounter 3	0	containers
seed of uniform	67	-
seed of arrivaltimes feeders	17	-
seed of arrivaltimes big	27	-
import generator feeders	0	containers
export generator feeders	0	containers
import generator big vessels	0	containers
export generator big vessels	0	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
rgv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	70	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	58	seconds

Table 19.1: Input parameters for experiment: shuffling from Berth 2

Parameter	Value	
frombtob[1,2]	8511	containers
frombtob[3,2]	8439	containers
frombtob[2,1]	0	containers
frombtob[2,3]	0	containers
Inplacecounter 1	1533	containers
Inplacecounter 2	3150	containers
Inplacecounter 3	1617	containers

Table 19.2, output after experiment: shuffling from Berth 2

In Table 19.2 the parameters inplacecounter 1,2 and 3 indicate 6300 containers in total have been shuffled between the yards.

19.1.1 crane occupancy rates

The cranes located in Yard 2 will have to deal with 6300 containers. 3150 containers to be delivered to Yard 1 & 3 from Yard 2, while 3150 containers have to be delivered from Yard 2 to Yard 1 & 3.

Considering that the number of containers that will be retrieved and delivered form the 20' stack and the 40' stacks are the same means that 3150 containers will be handled by both stacks types located on Yard 2.

- Required 20' warehouse crane operational time: 3150 * 80 seconds = 252,000 seconds
- Required 40' warehouse crane operational time: 3150 * 80 seconds = 252,000 seconds

Number of warehouse cranes in Yard 2:
- Number of warehouse cranes in the 20' stacks in Yard 2 = 6 * 2 = 12 warehouse cranes
- Number of warehouse cranes in the 40' stacks in Yard 2 = 6 * 3 = 18 warehouse cranes

Needed operational time per warehouse crane:

- $252{,}000/12 = 21{,}000$ seconds per 20' stack warehouse crane
- $252{,}000/18 = 14{,}000$ seconds per 40' stack warehouse crane

Required adjacent loading time for both feeder vessels: 11.25 hours $\Rightarrow$ 40,500 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(21{,}000/40{,}500\right) * 100\% = 51.9\ \%$

40' stack: $\left(14{,}000/40{,}500\right) * 100\% = 34.6\ \%$

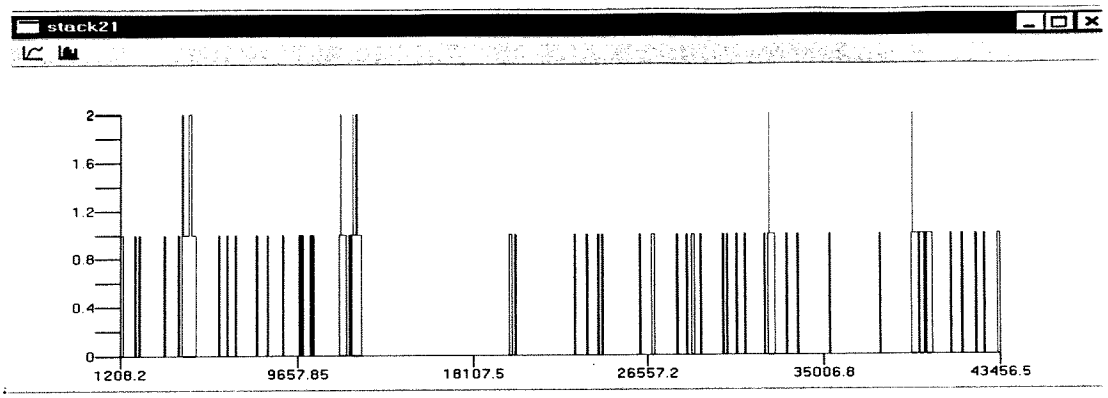


Figure 19.1, rgv queues in warehouse lane 21 when shuffling from Berth 2

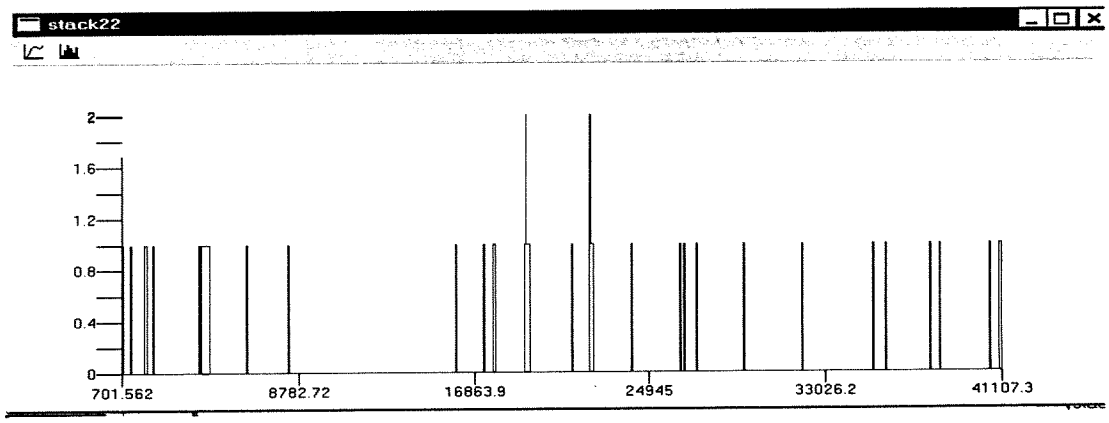


Figure 19.2, rgv queues in warehouse lane 22 when shuffling from Berth 2

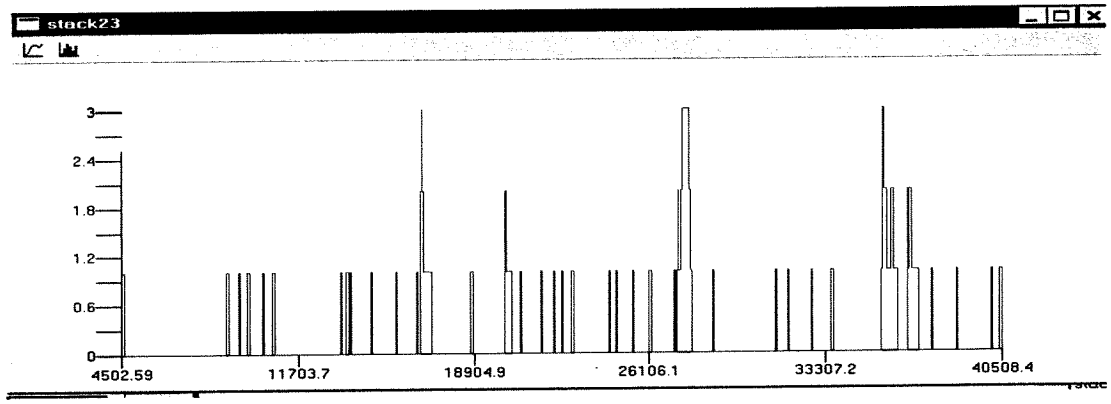


Figure 19.3, rgv queues in warehouse lane 23 when shuffling from Berth 2

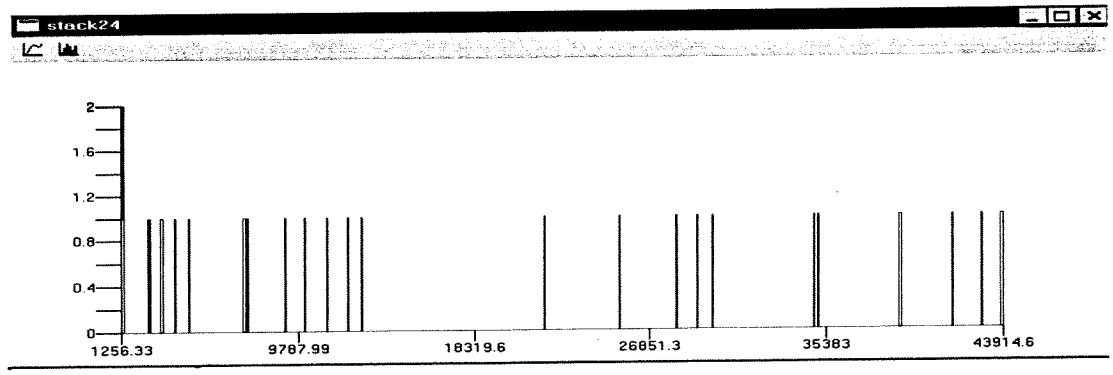


Figure 19.4, rgv queues in warehouse lane 24 when shuffling from Berth 2

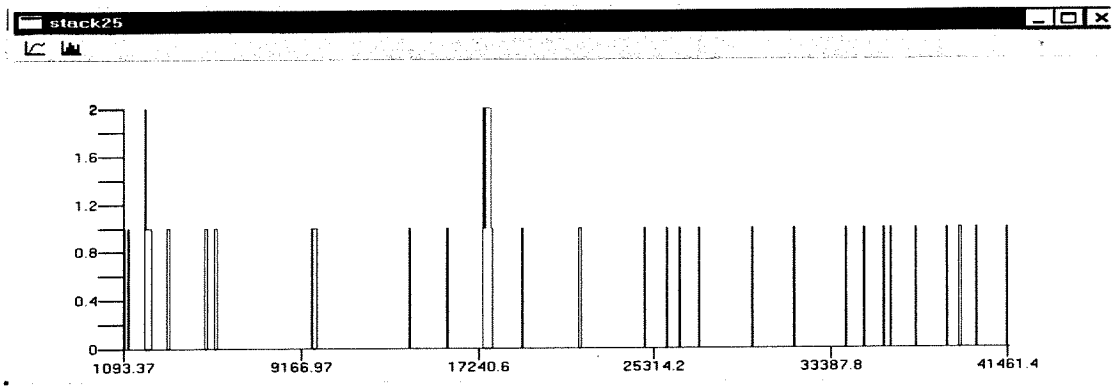


Figure 19.5, rgv queues in warehouse lane 25 when shuffling from Berth 2

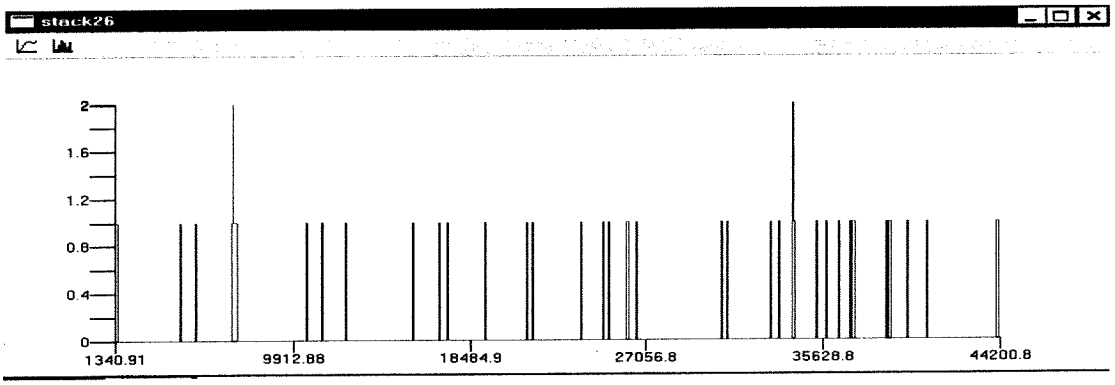


Figure 19.6, rgv queues in warehouse lane 26 when shuffling from Berth 2

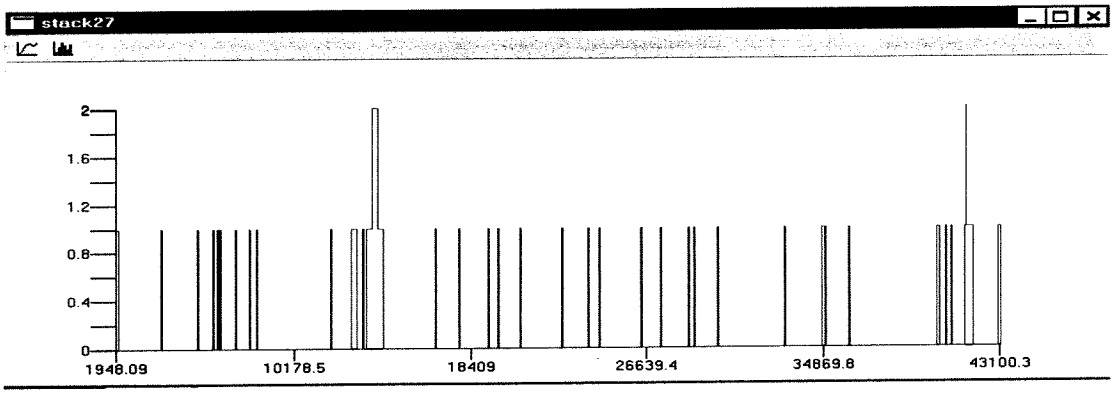


Figure 19.7, rgv queues in warehouse lane 27 when shuffling from Berth 2

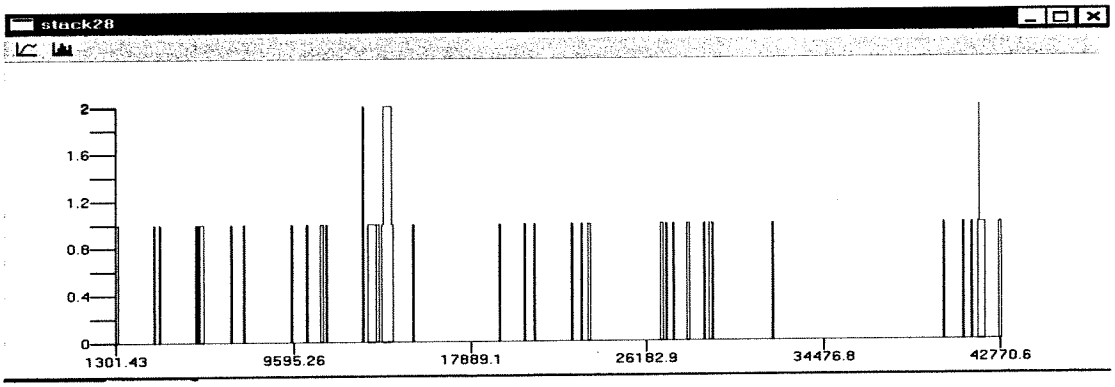


Figure 19.8, rgv queues in warehouse lane 28 when shuffling from Berth 2

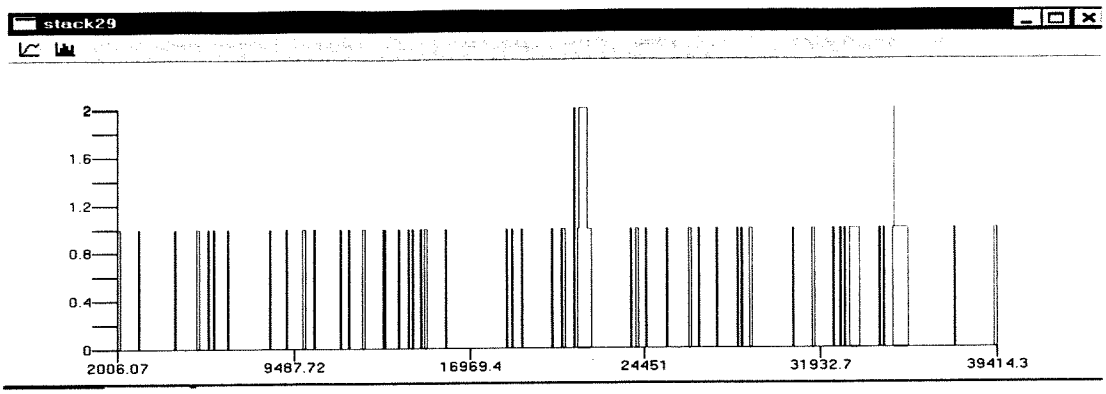


Figure 19.9, rgv queues in warehouse lane 29 when shuffling from Berth 2

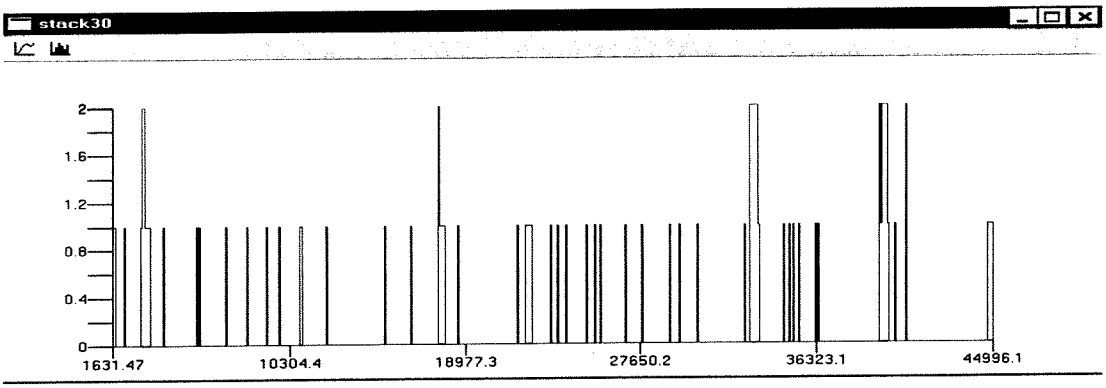


Figure 19.10, rgv queues in warehouse lane 30 when shuffling from Berth 2

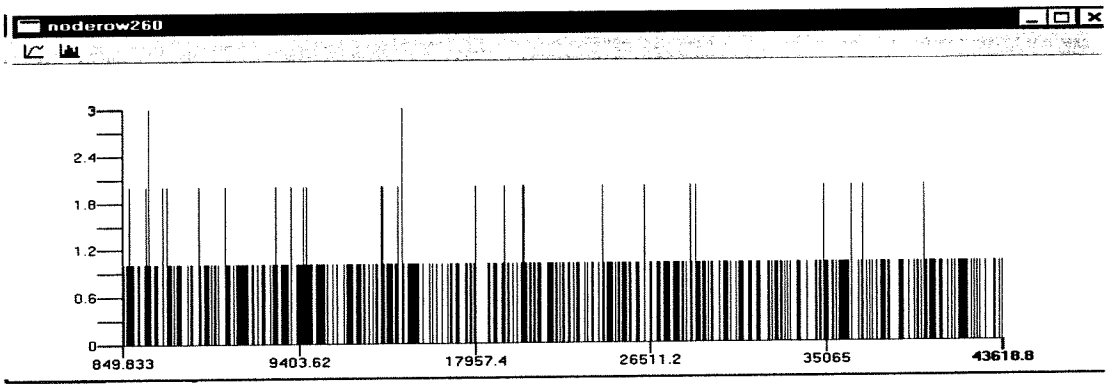


Figure 19.11, rgv waiting queue at node 260

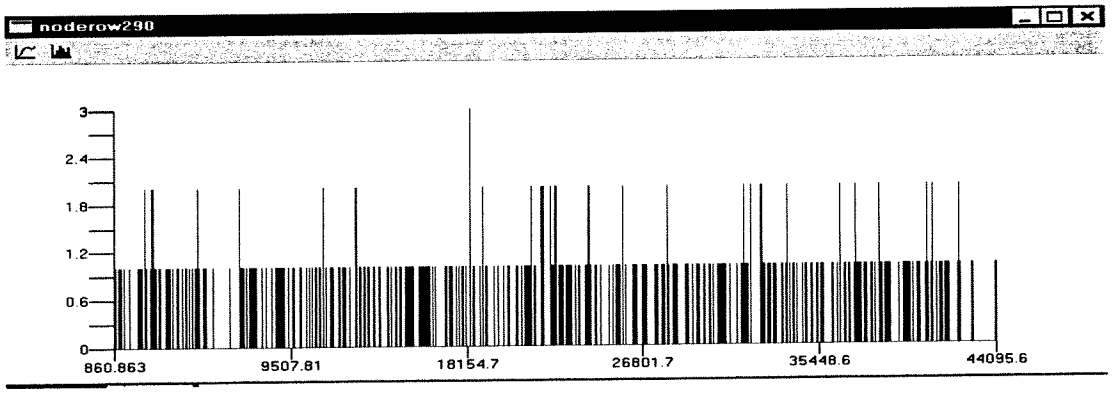


Figure 19.12, rgv waiting queue at node 290

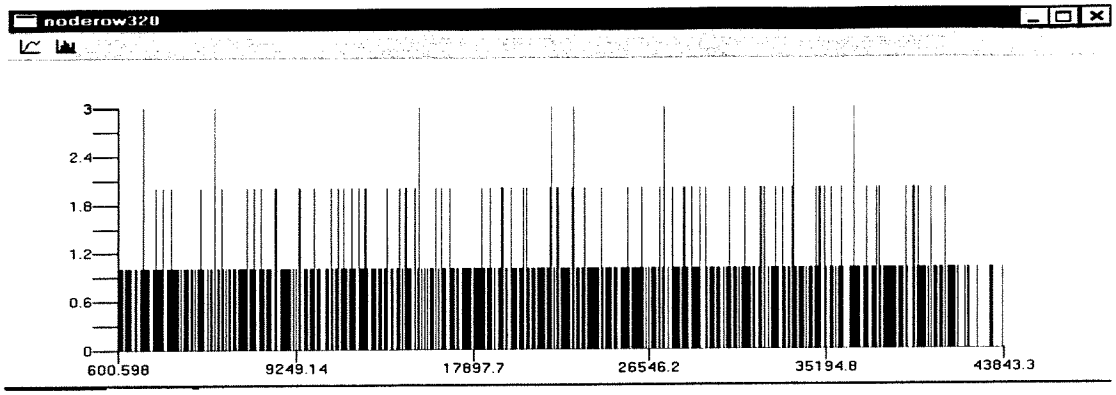


Figure 19.13, rgv waiting queue at node 320

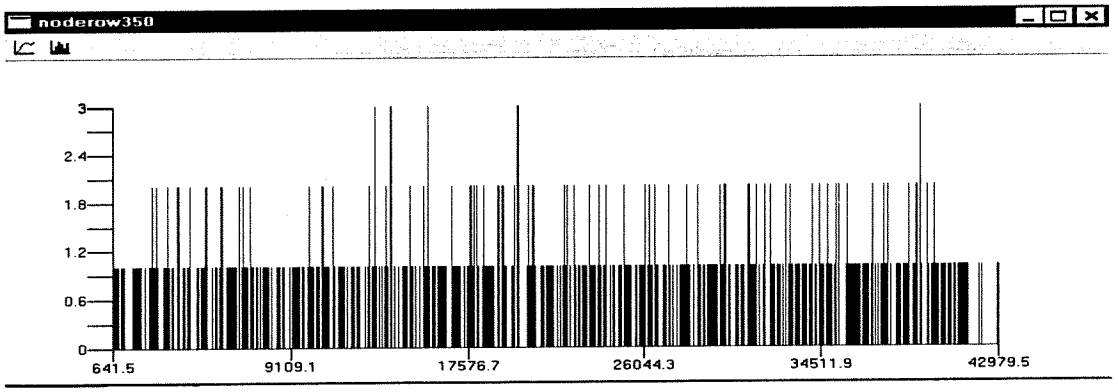


Figure 19.14, rgv waiting queue at node 350

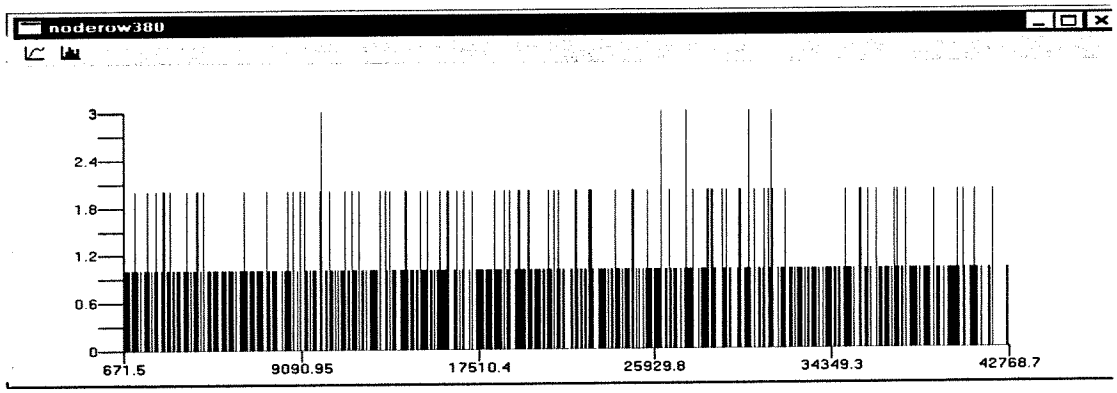


Figure 19.15, rgv waiting queue at node 380

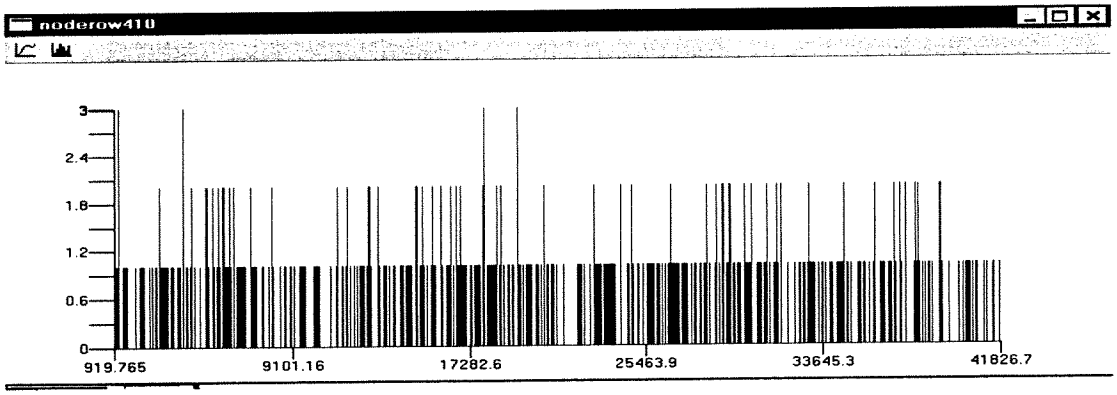


Figure 19.16, rgv waiting queue at node 410

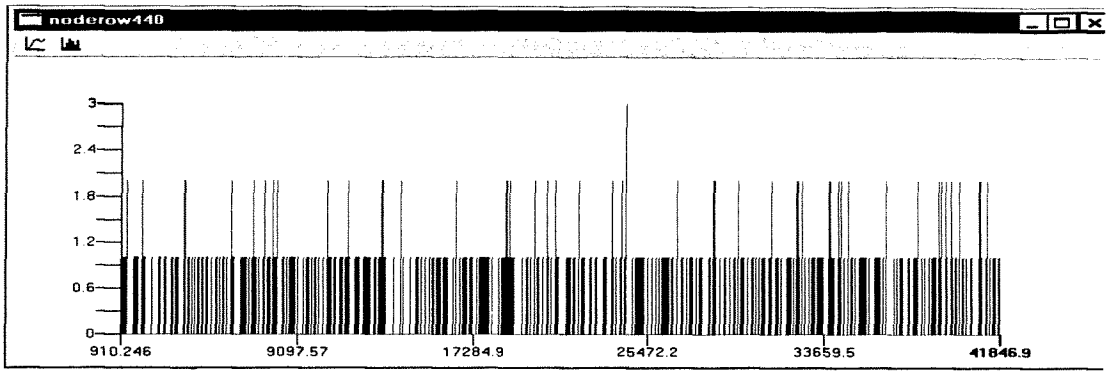


Figure 19.17, rgv waiting queue at node 440

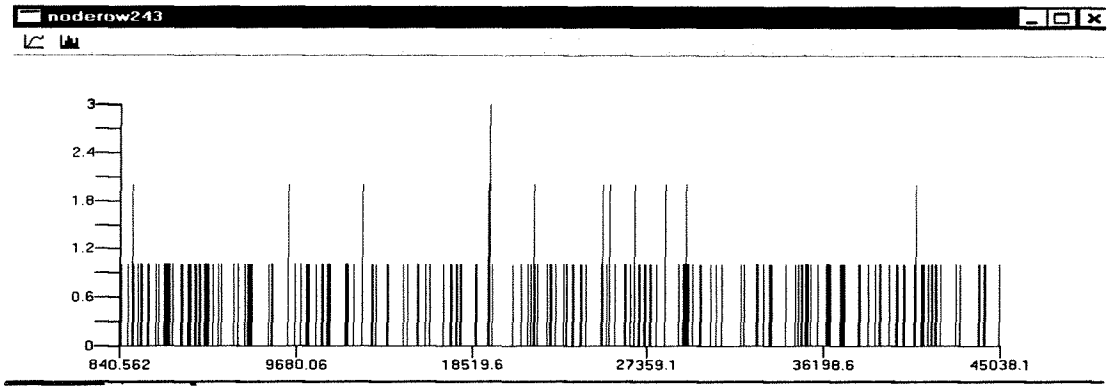


Figure 19.18, rgv waiting queue at node 243

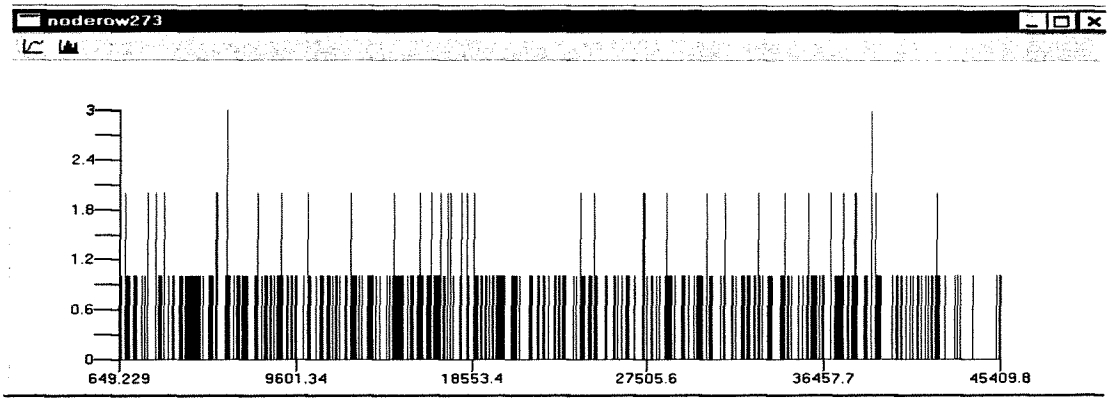


Figure 19.19, rgv waiting queue at node 273

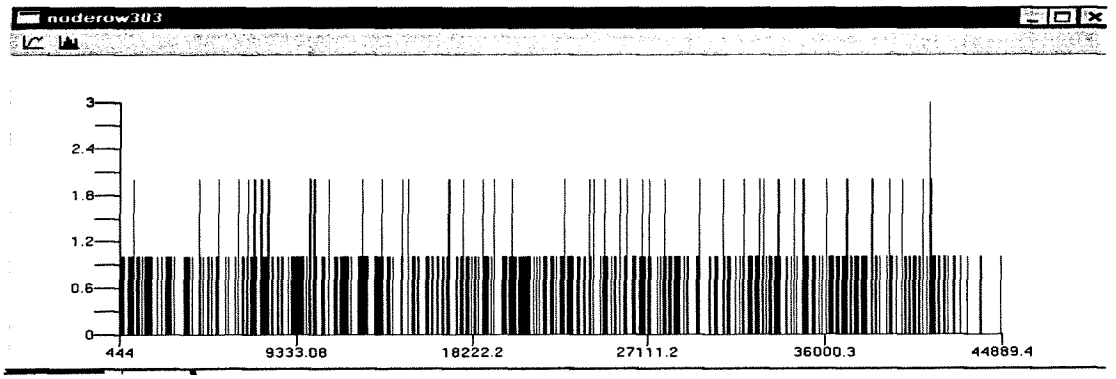


Figure 19.20, rgv waiting queue at node 303

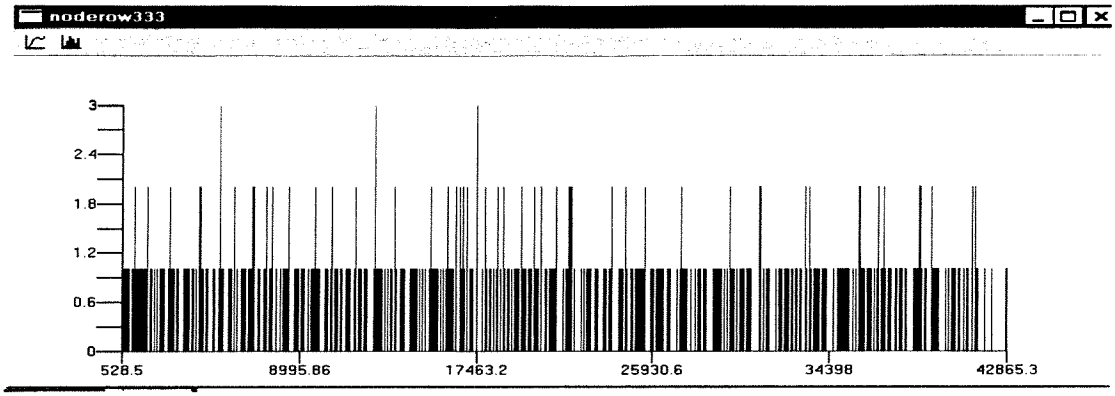


Figure 19.21, rgv waiting queue at node 333

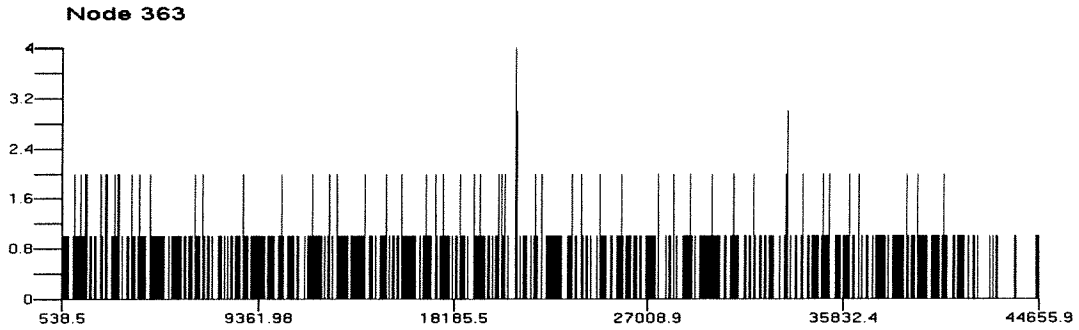


Figure 19.22, rgv waiting queue at node 363

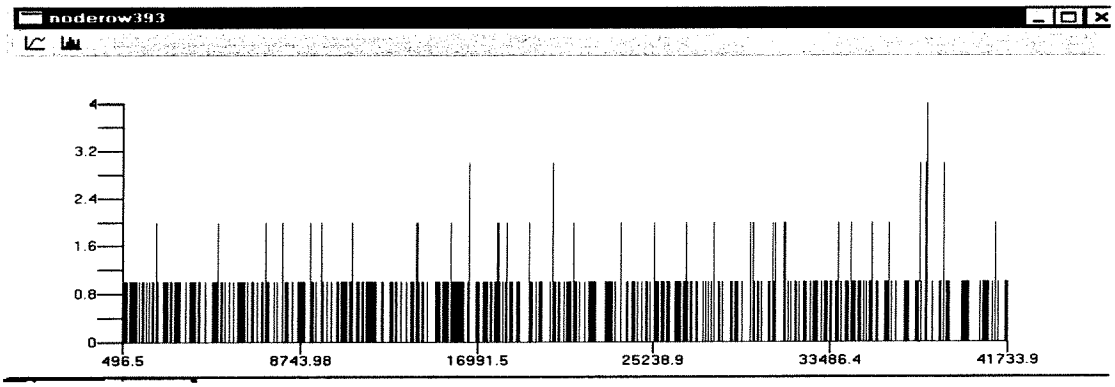


Figure 19.23, rgv waiting queue at node 393

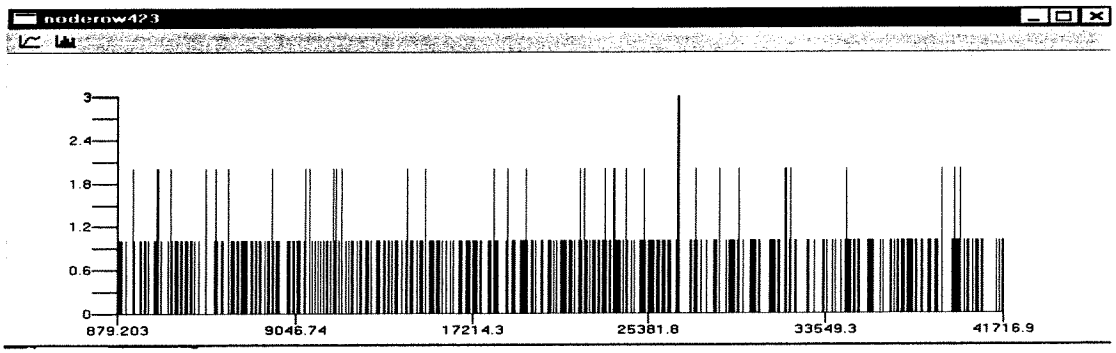


Figure 19.24, rgv waiting queue at node 423

Appendix 20: Input parameters for the experiment adjacent loading at all three berths

The basic parameters used for the experiment adjacent loading from all three berths are shown in Table 20.1. The import generators of the feeder and the mega vessel only produce a few containers while this is not important in this experiment. The parameters frombtob[1,2] represent the containers in Yard 1 which have to be loaded unto a vessel at Berth 2.
The changed parameters after the run are shown in Table 20.2

Parameter	Value	
frombtob[1,2]	2500	containers
frombtob[3,2]	2500	containers
frombtob[2,1]	1500	containers
frombtob[2,3]	1500	containers
inplacecounter 1	0	containers
inplacecounter 2	0	containers
inplacecounter 3	0	containers
seed of uniform	67	-
seed of arrivaltimes feeders	17	-
seed of arrivaltimes big	27	-
import generator feeders	80	containers
export generator feeders	1050	containers
import generator big vessels	120	containers
export generator big vessels	3150	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
rgv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	70	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	58	seconds

Table 20.1: Input parameters for experiment: adjacent loading at all three berths

Parameter	Value	
frombtob[1,2]	1018	containers
frombtob[3,2]	912	containers
frombtob[2,1]	485	containers
frombtob[2,3]	475	containers

Table 20.2, output after experiment adjacent loading at all 3 berths

20.1.1 Warehouse cranes located in Yard 2 while adjacent loading on all three the berths

As both feeder vessels will be provided from Yard 2 this means that 1050 * 2 = 2100 containers have to retrieved from these stacks. Considering that the number of containers that will retrieved from the 20' stacks and the 40' stacks are the same means that 1050 will be retrieved from both stack types.

Required 20' warehouse crane operational time: 1050 * 80 seconds = 84,000 seconds
Required 40' warehouse crane operational time: 1050 * 80 seconds = 84,000 seconds

Number of warehouse cranes in Yard 2:
Number of warehouse cranes in the 20' stacks in Yard 2 = 6 * 2 = 12 warehouse cranes
Number of warehouse cranes in the 40' stacks in Yard 2 = 6 * 3 = 18 warehouse cranes

Needed operational time per warehouse crane:

$84{,}000/12 = 7{,}000$ seconds per 20' stack warehouse crane

$84{,}000/18 = 4{,}667$ seconds per 40' stack warehouse crane

Required adjacent loading time for both feeder vessels: 5.4 hours $\Rightarrow$ 20160 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(7{,}000/20160\right) * 100\% = 34.7\,\%$

40' stack: $\left(4{,}667/20160\right) * 100\% = 23.2\,\%$

In the same manner the occupancy rates of the warehouse cranes located in Yard 1 and 3 have been calculated. In 5.4 hours 1752 containers have been handled.

Required 20' warehouse crane operational time: 876 * 80 seconds = 70,080 seconds
Required 40' warehouse crane operational time: 876 * 80 seconds = 70,080 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(8{,}760/20{,}160\right) * 100\% = 43.5\,\%$

40' stack: $\left(5{,}840/20{,}160\right) * 100\% = 29.0\,\%$

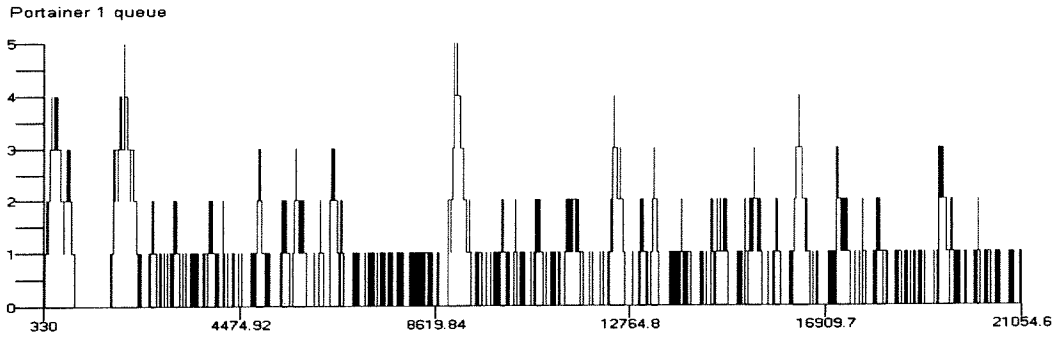


Figure 20.1 rgv queue under portainer 1

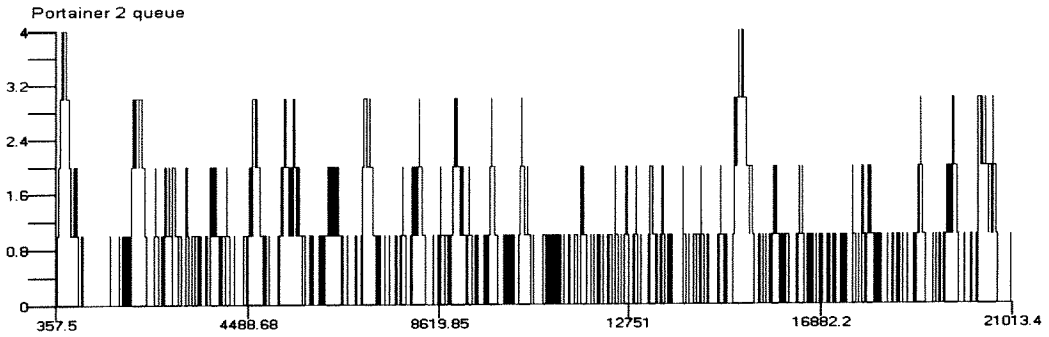


Figure 20.2 rgv queue under portainer 2

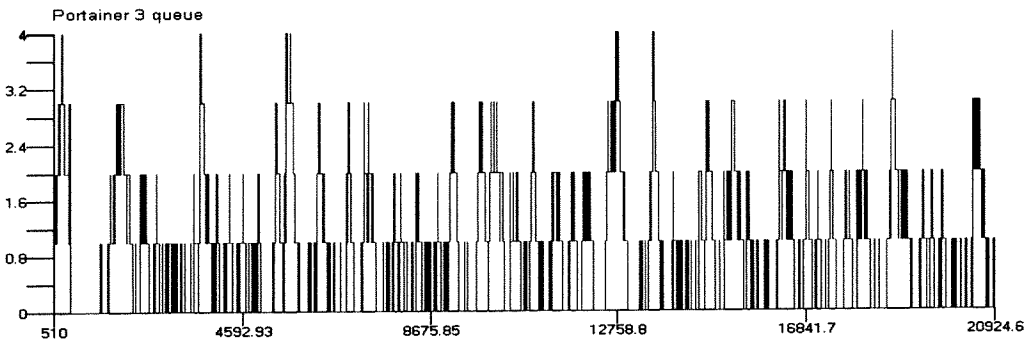


Figure 20.3 rgv queue under portainer 3

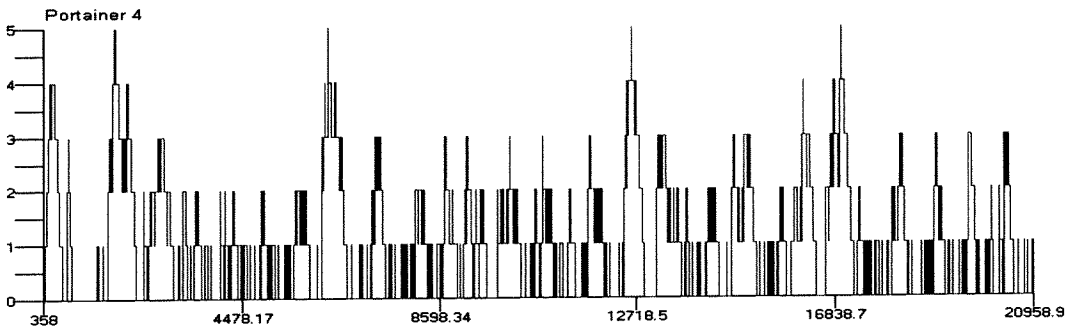


Figure 20.4, rgv queue under portainer 4

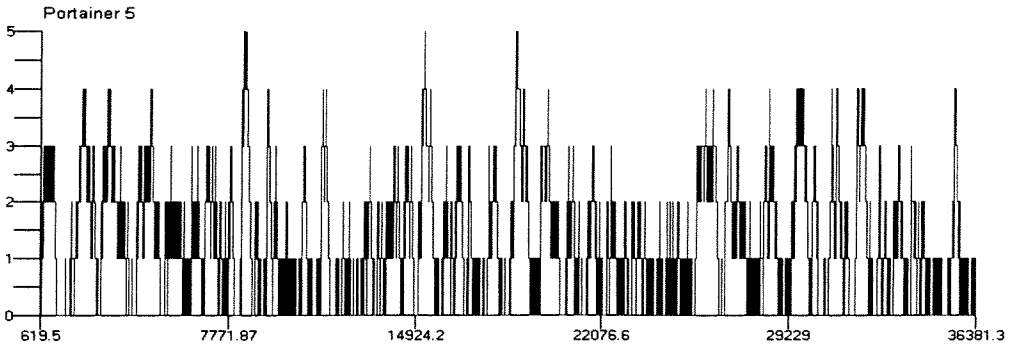


Figure 20.5 rgv queue under portainer 5

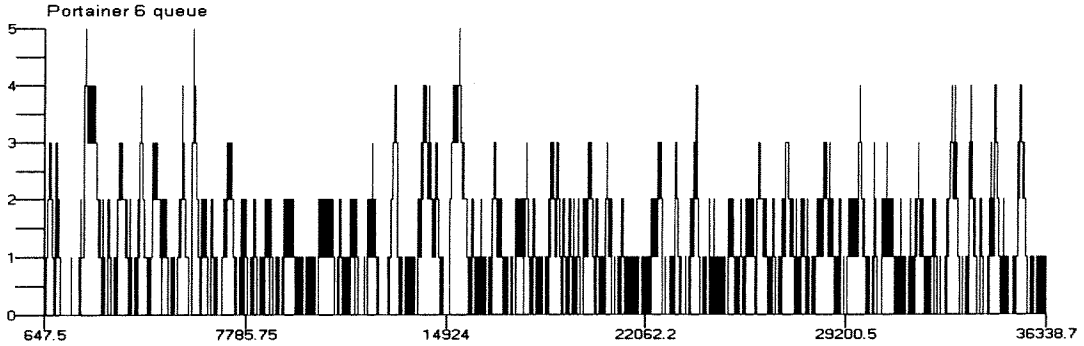


Figure 20.6 rgv queue under portainer 7

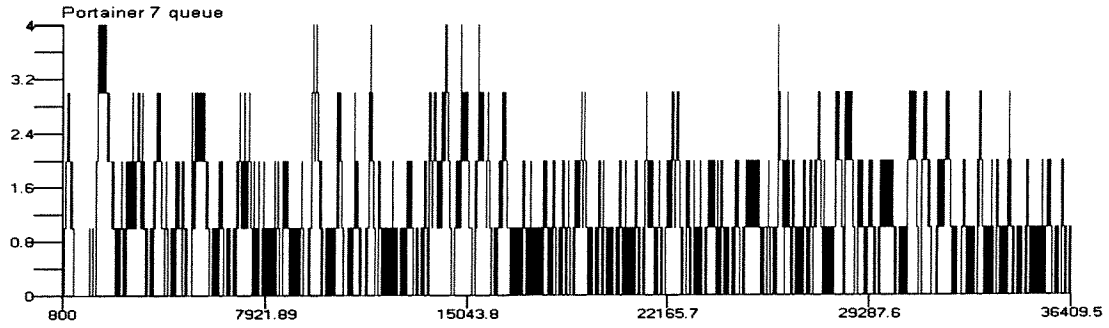


Figure 20.7 rgv queue under portainer 7

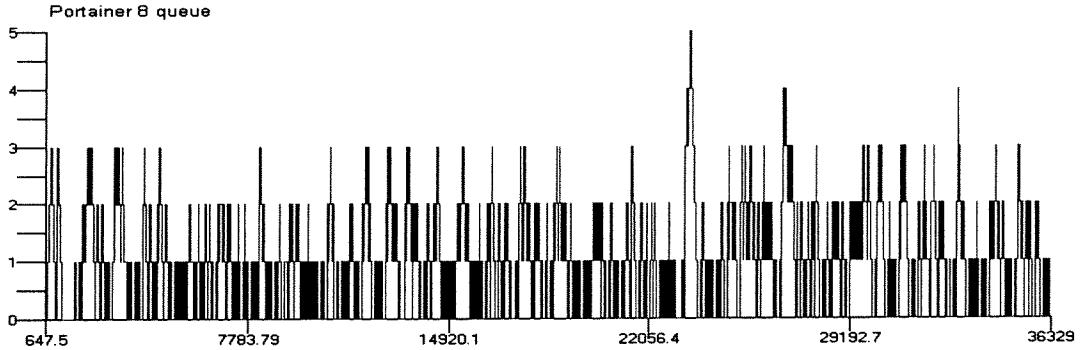


Figure 20.8 rgv queue under portainer 8

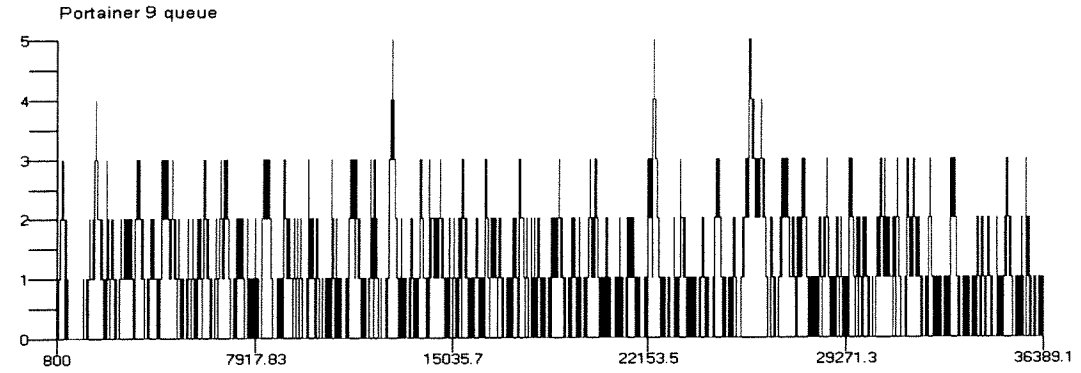


Figure 20.9 rgv queue under portainer 9

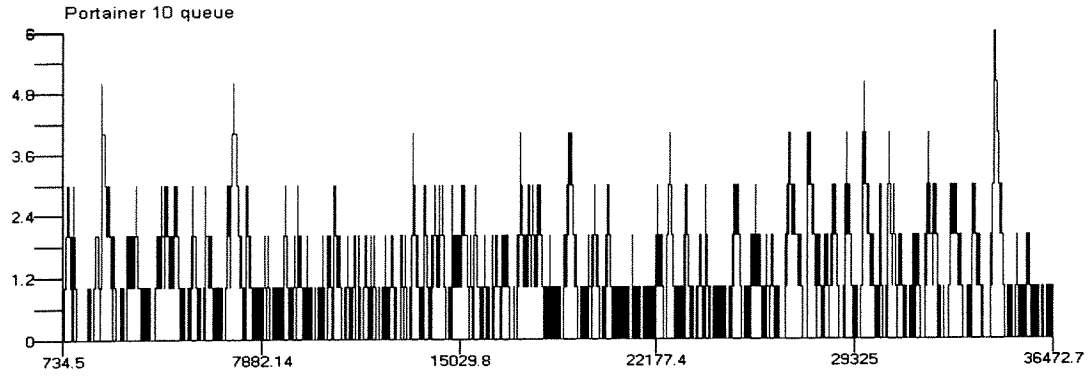


Figure 20.10 rgv queue under portainer 10

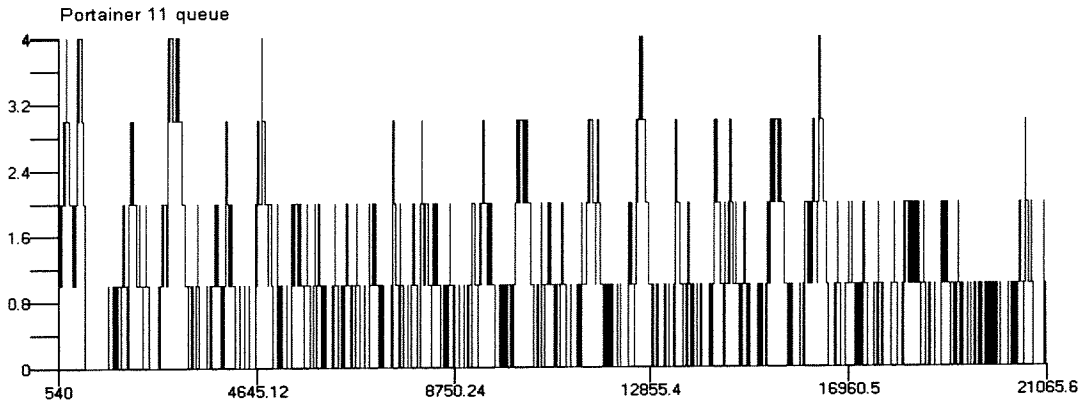


Figure 20.11 rgv queue under portainer 11

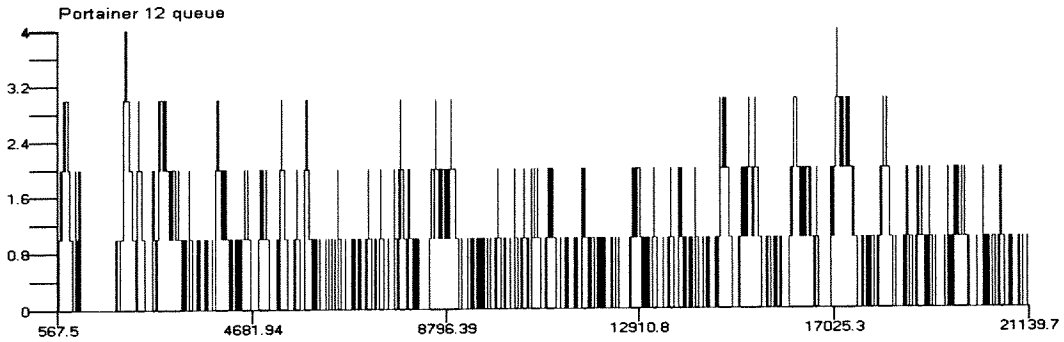


Figure 20.12 rgv queue under portainer 12

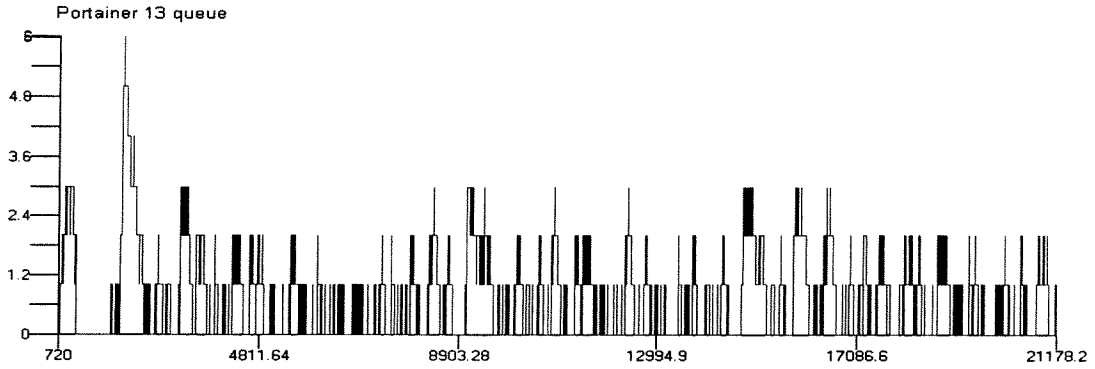


Figure 20.13 rgv queue under portainer 13

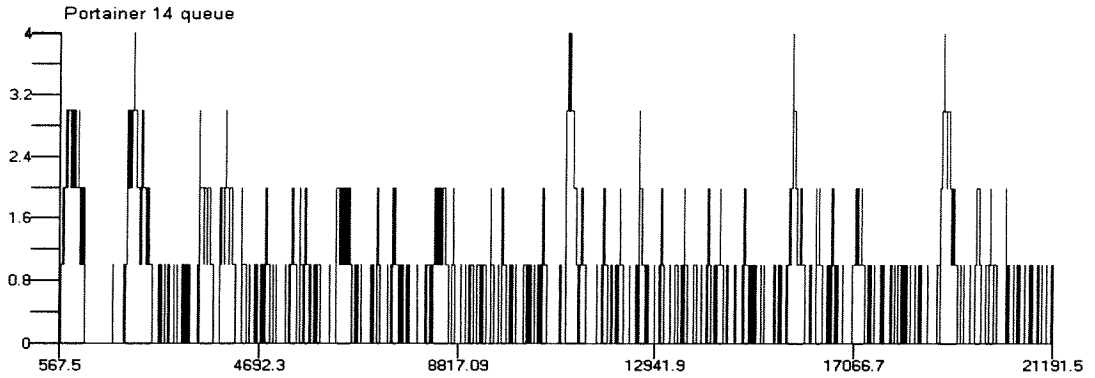


Figure 20.14 rgv queue under portainer 14

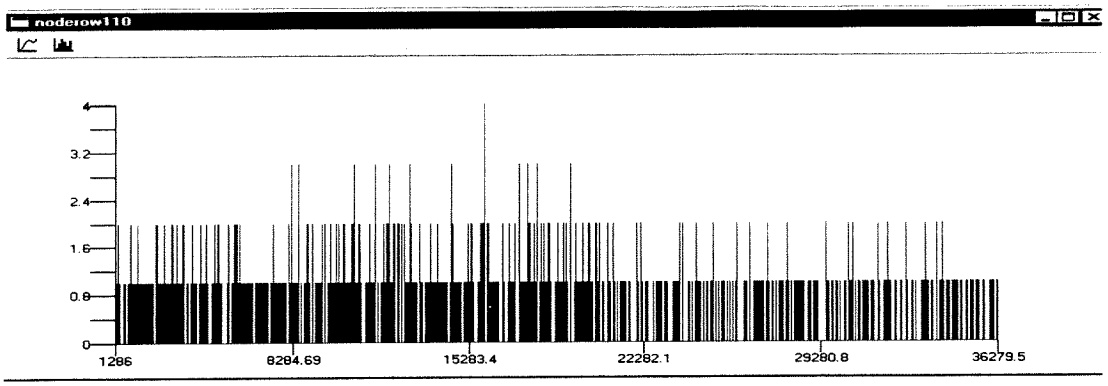


Figure 20.15 rgv queue at node 110

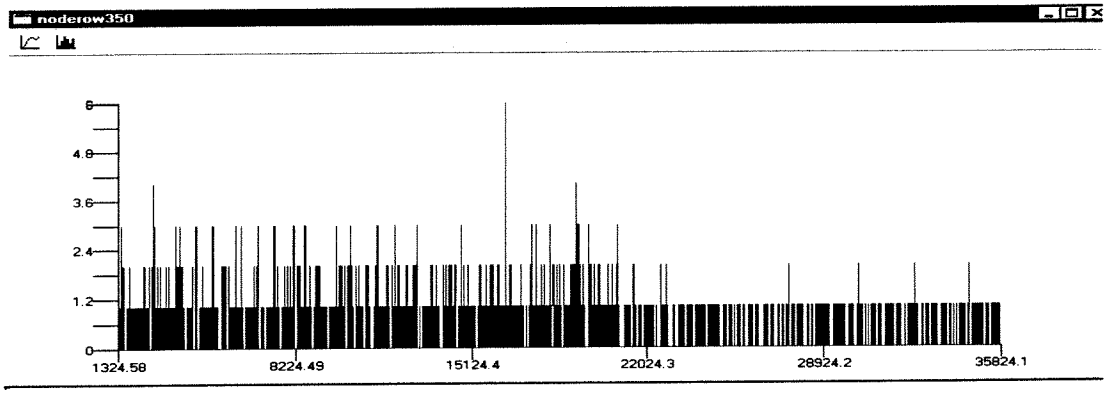


Figure 20.16 rgv queue at node 350

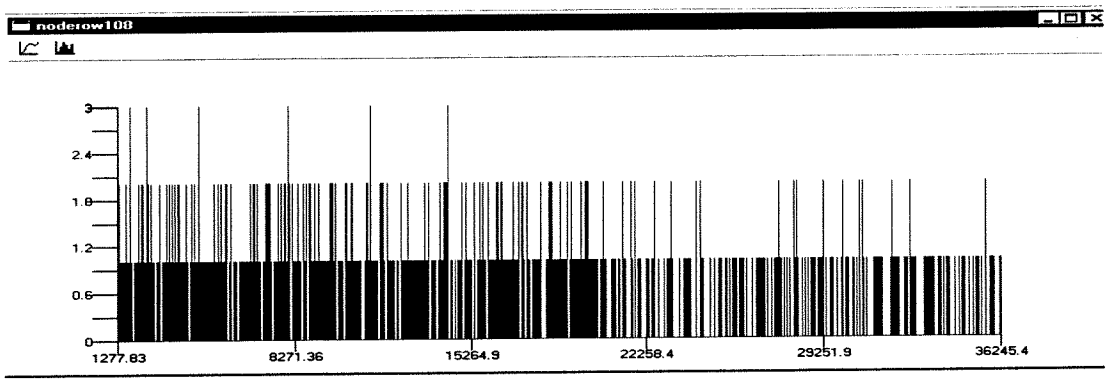


Figure 20.17 rgv queue at node 108

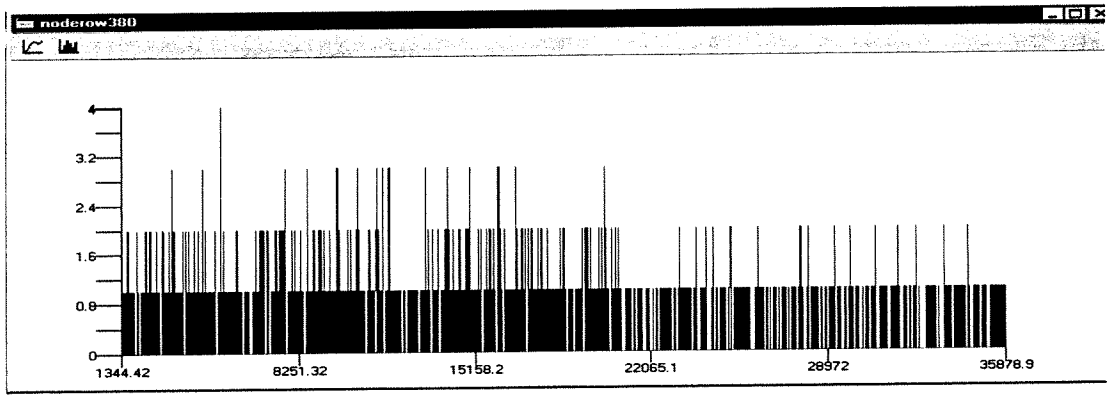


Figure 20.18 rgv queue at node 380

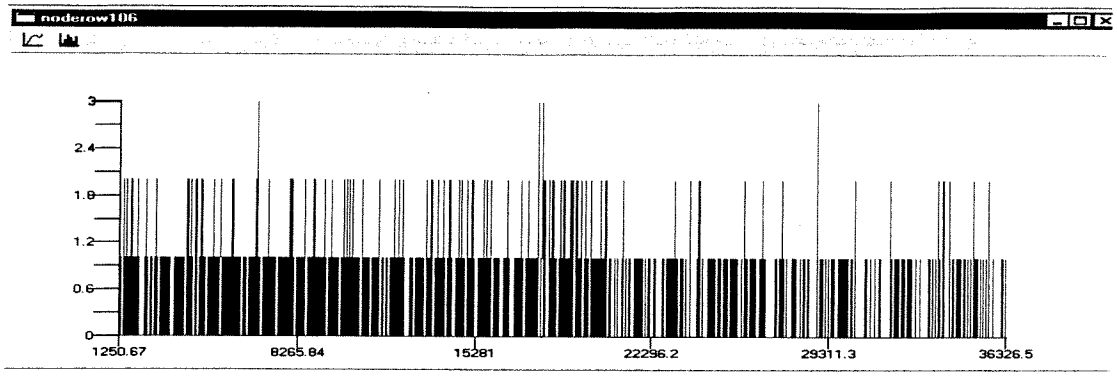


Figure 20.19 rgv queue at node 106

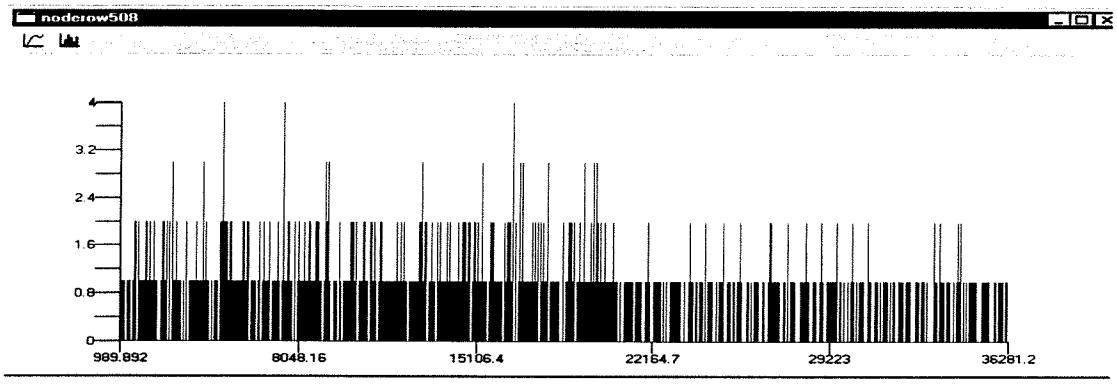


Figure 20.20 rgv queue at node 508

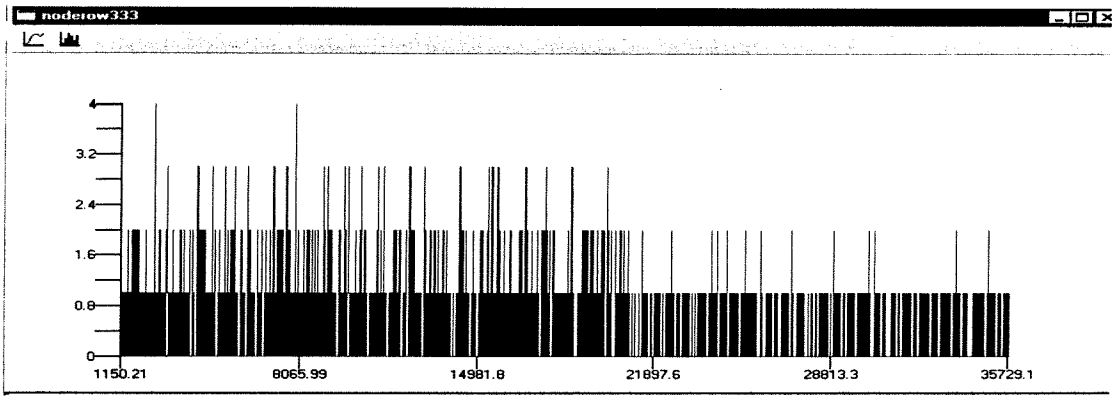


Figure 20.21 rgv queue at node 333

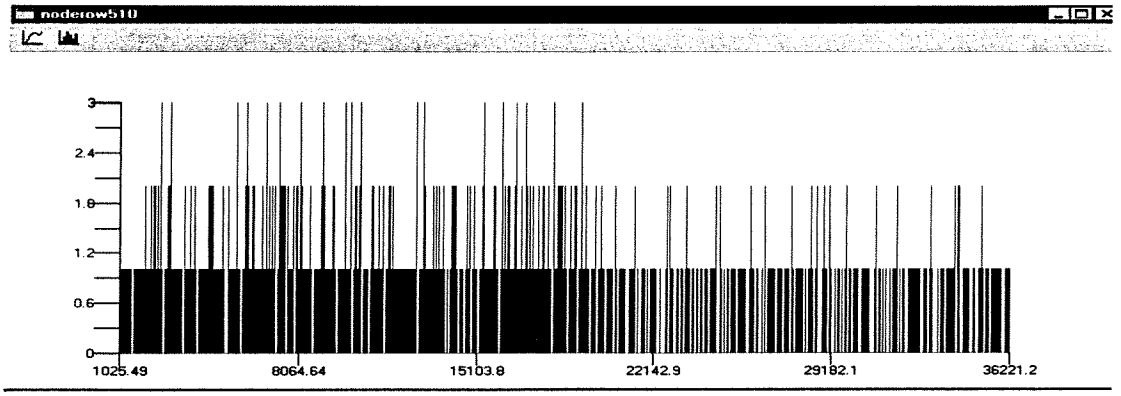


Figure 20.22, rgv queue at node 510

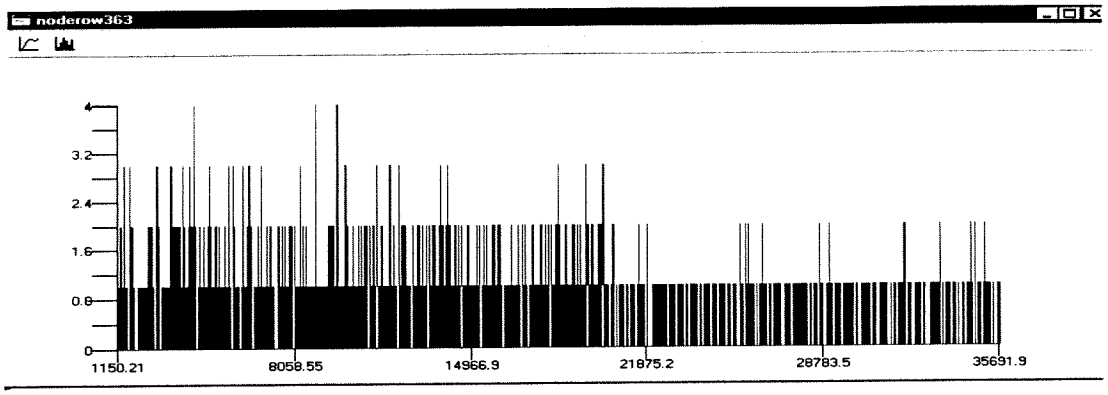


Figure 20.23 rgv queue at node 363

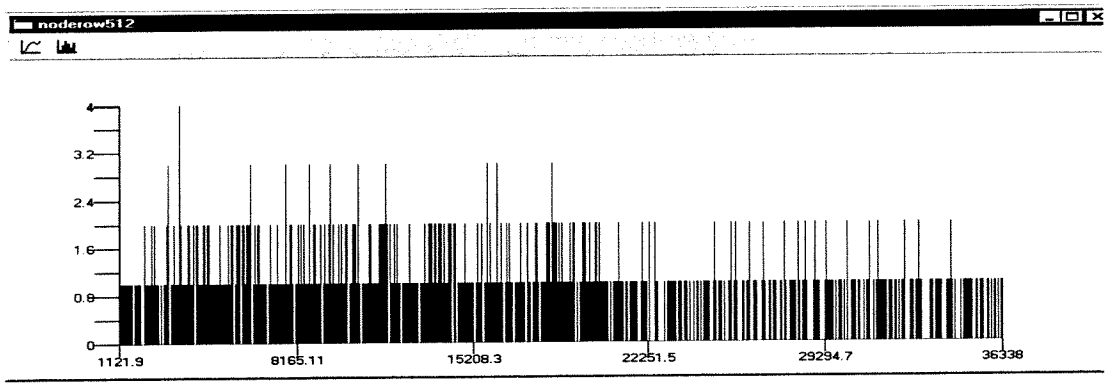


Figure 20.24 rgv queue at node 512

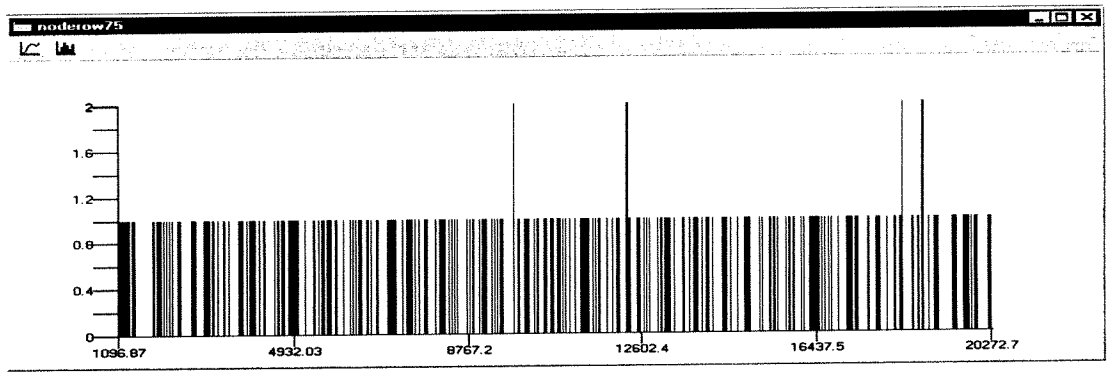


Figure 20.25 rgv queue at node 75

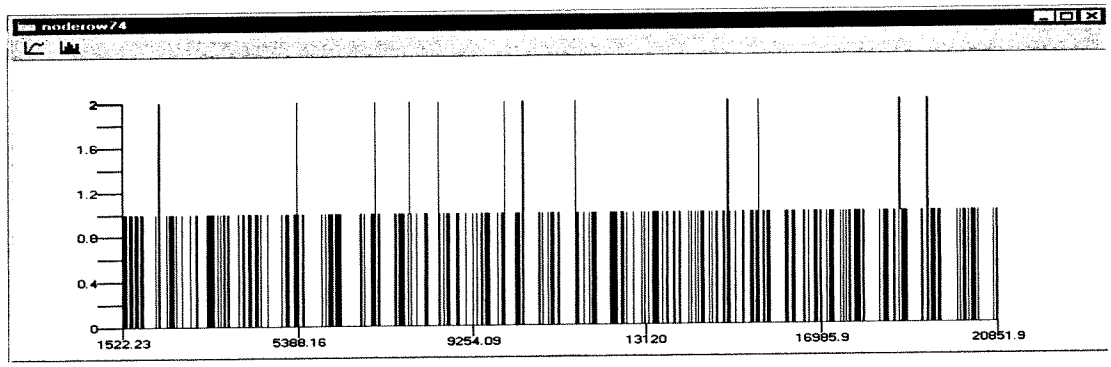


Figure 20.26 rgv queue at node 74

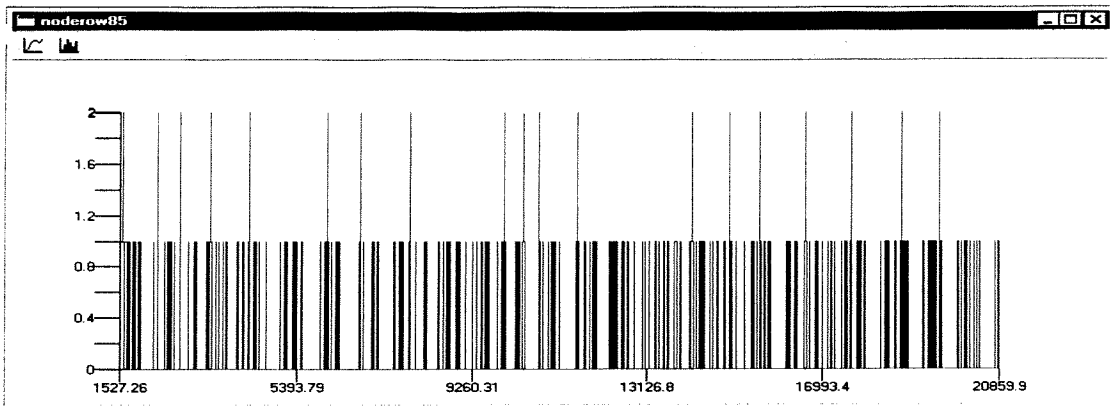


Figure 20.27 rgv queue at node 85

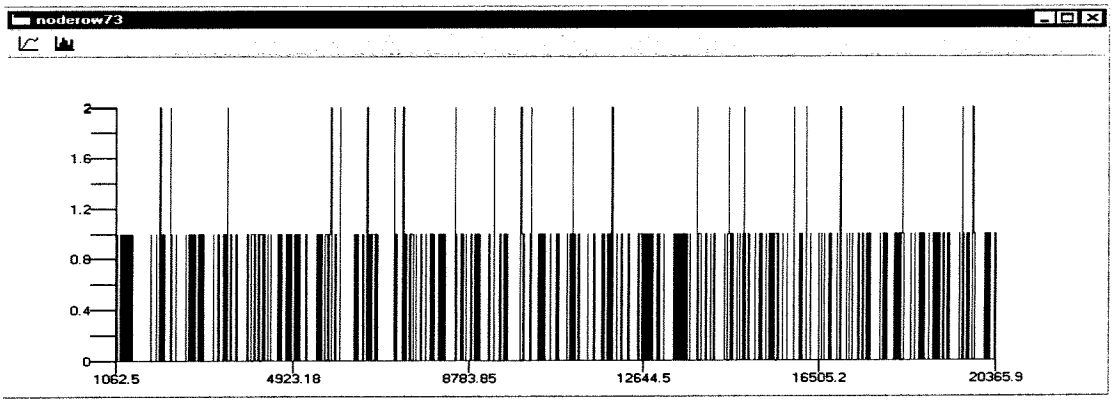


Figure 20.28 rgv queue at node 73

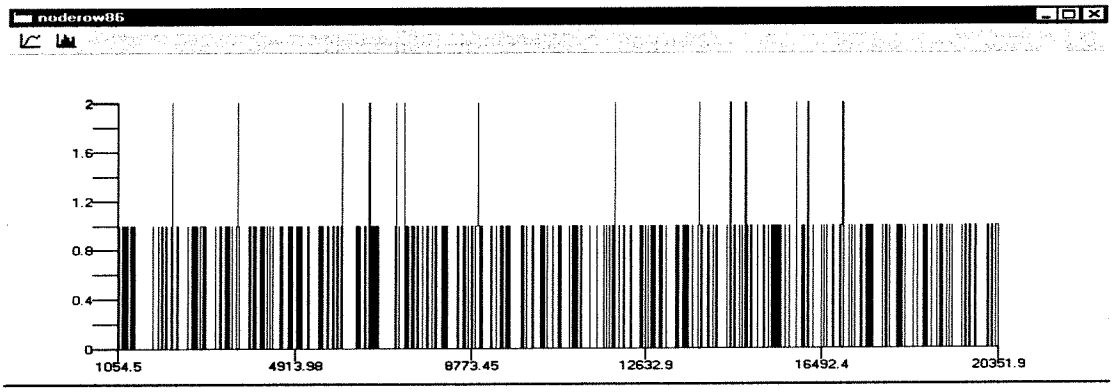


Figure 20.29 rgv queue at node 86

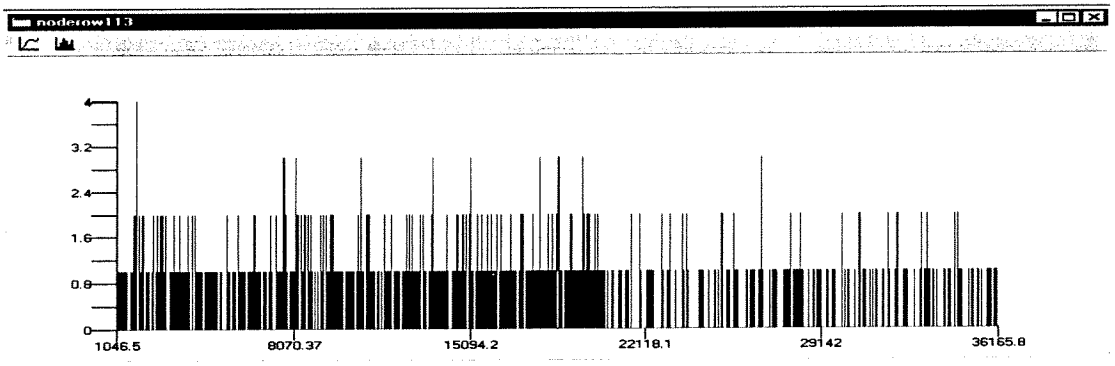


Figure 20.30 rgv queue at node 113

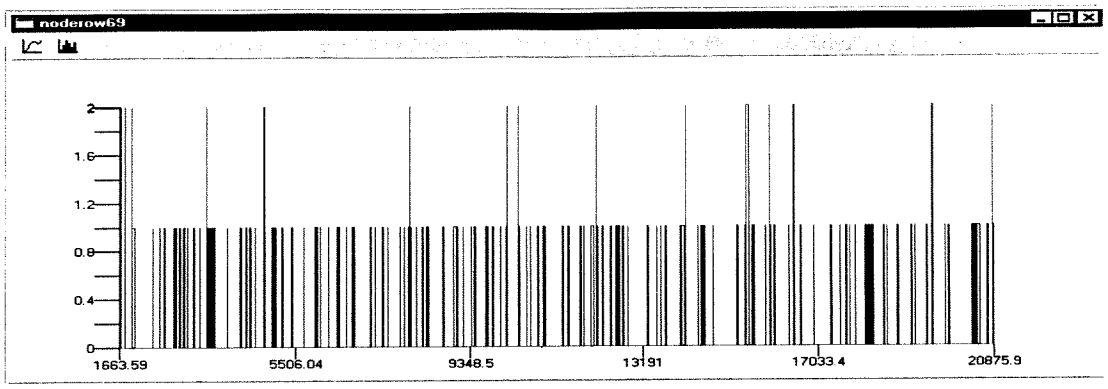


Figure 20.31 rgv queue at node 69

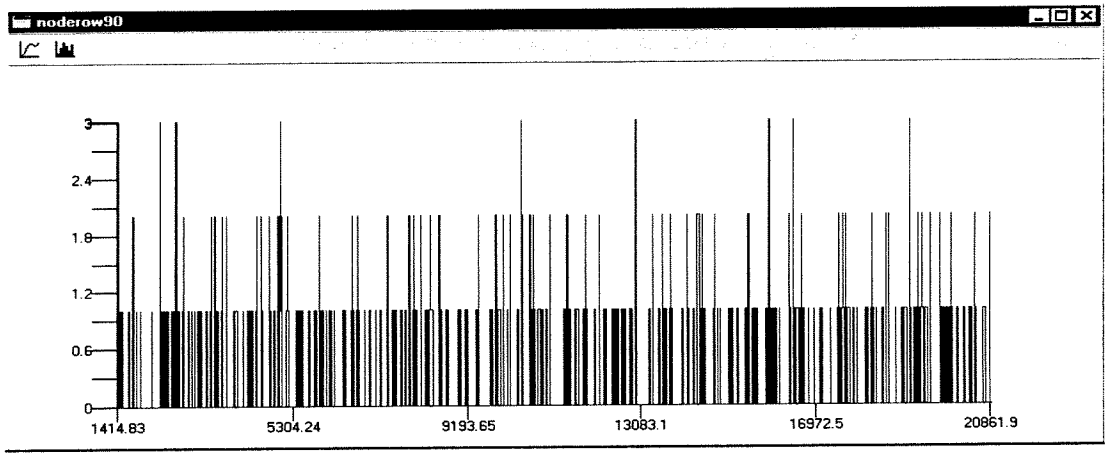


Figure 20.32 rgv queue at node 90

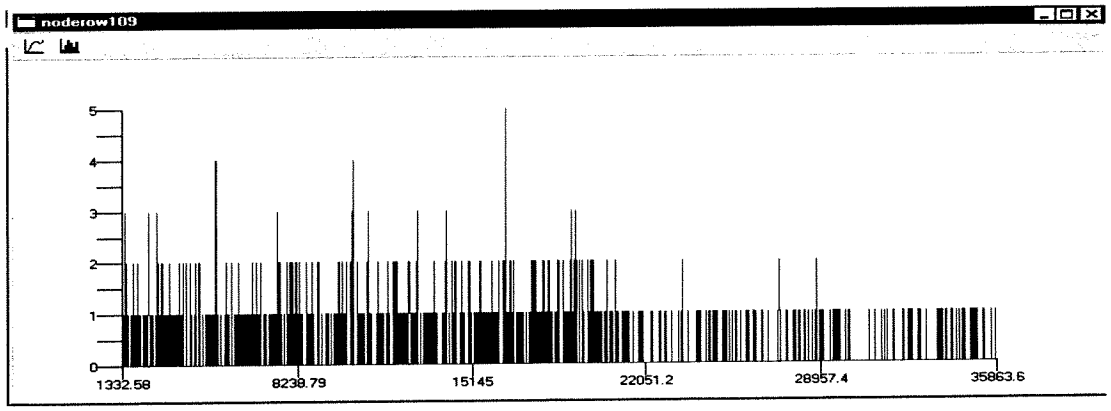


Figure 20.33 rgv queue at node 109

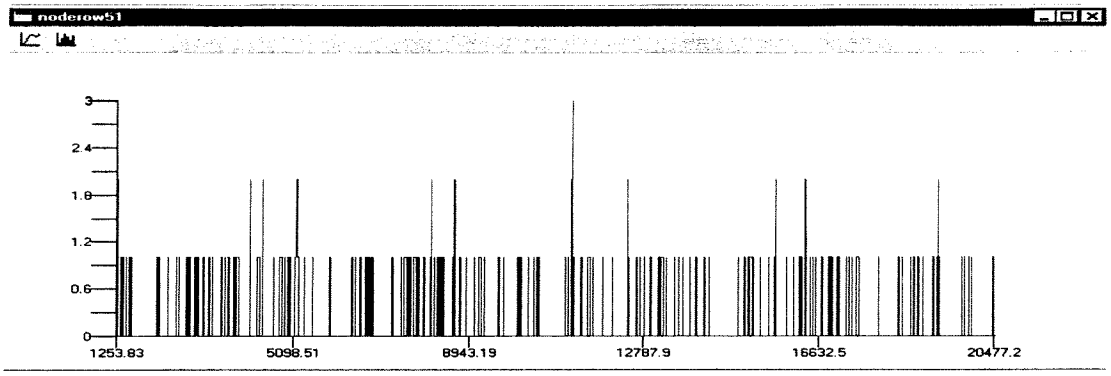


Figure 20.34 rgv queue at node 51

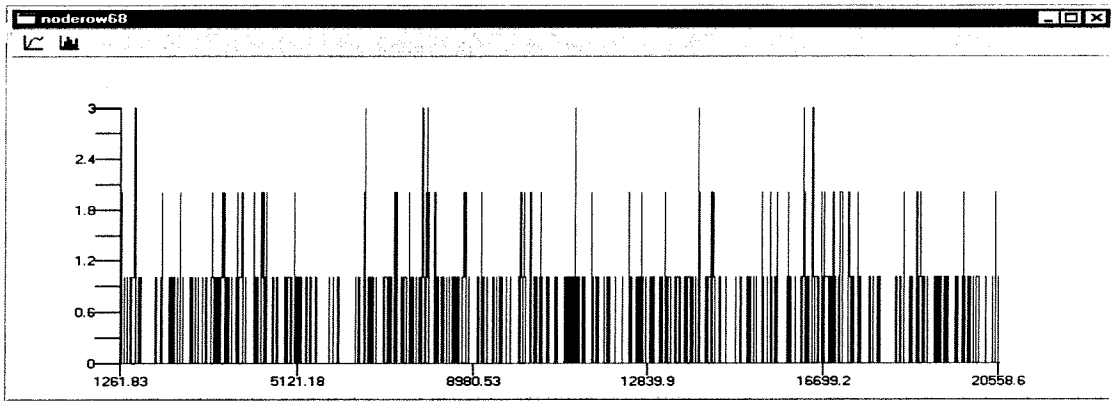


Figure 20.35 rgv queue at node 68

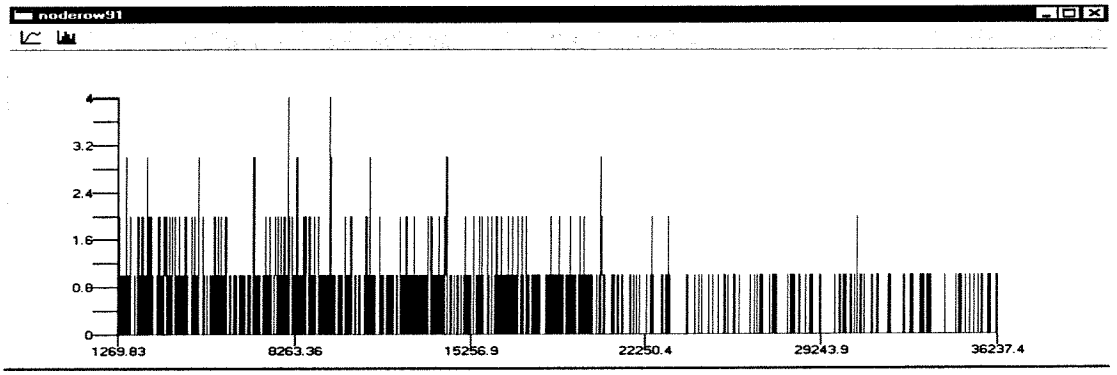


Figure 20.36 rgv queue at node 91

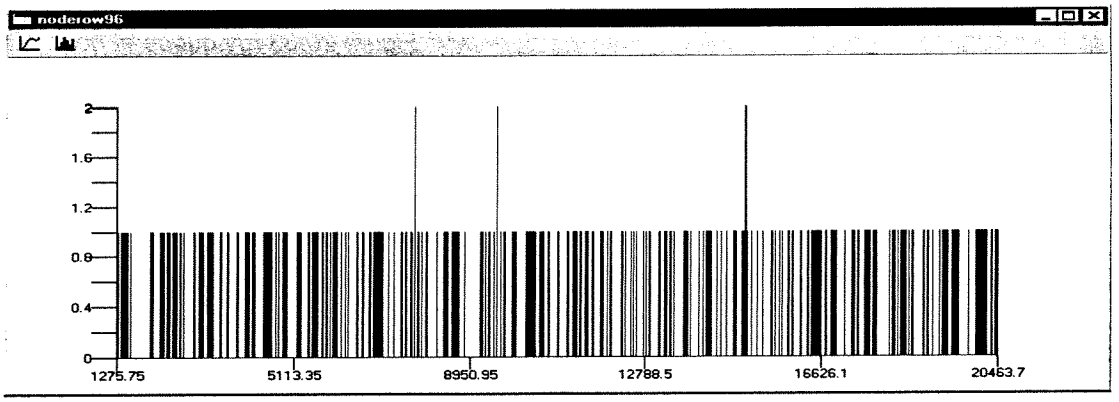


Figure 20.37 rgv queue at node 96

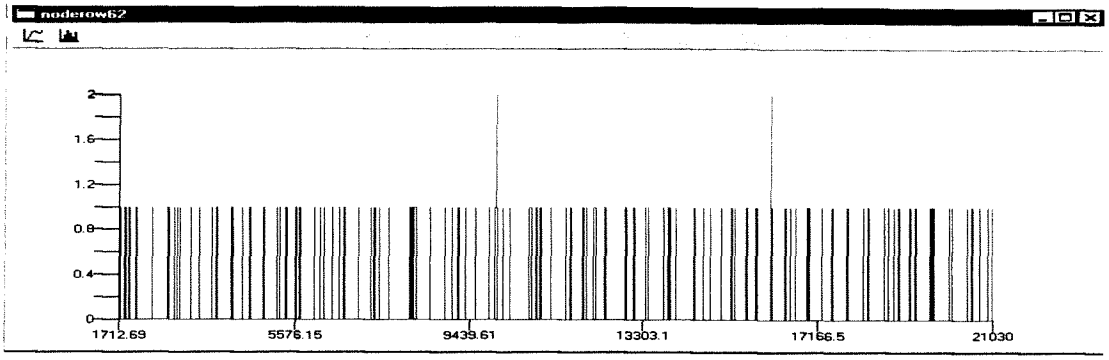


Figure 20.38 rgv queue at node 62

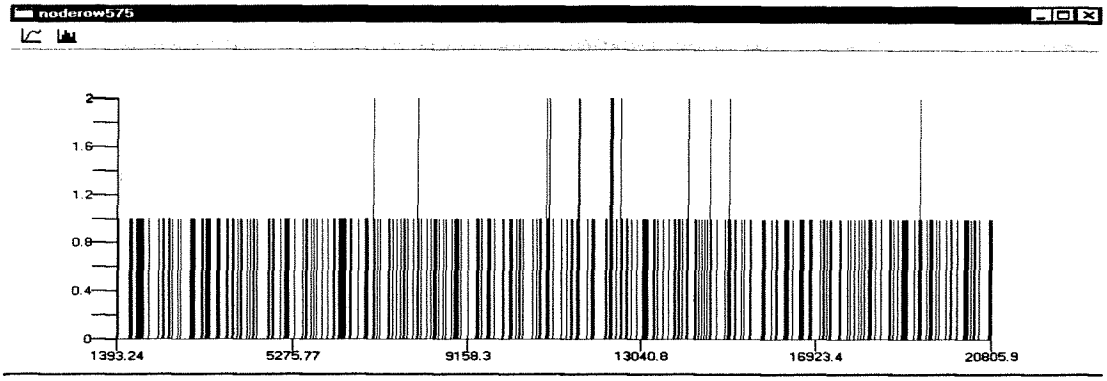


Figure 20.39 rgv queue at node

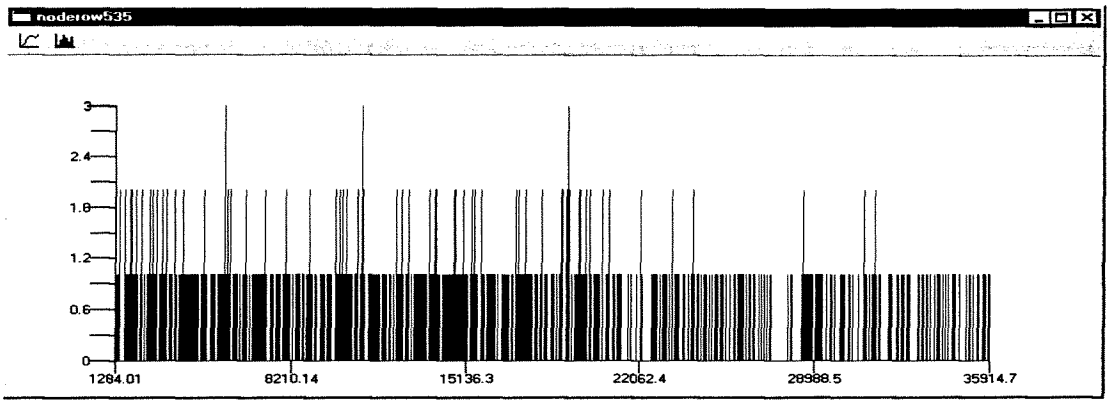


Figure 20.40 rgv queue at node 535

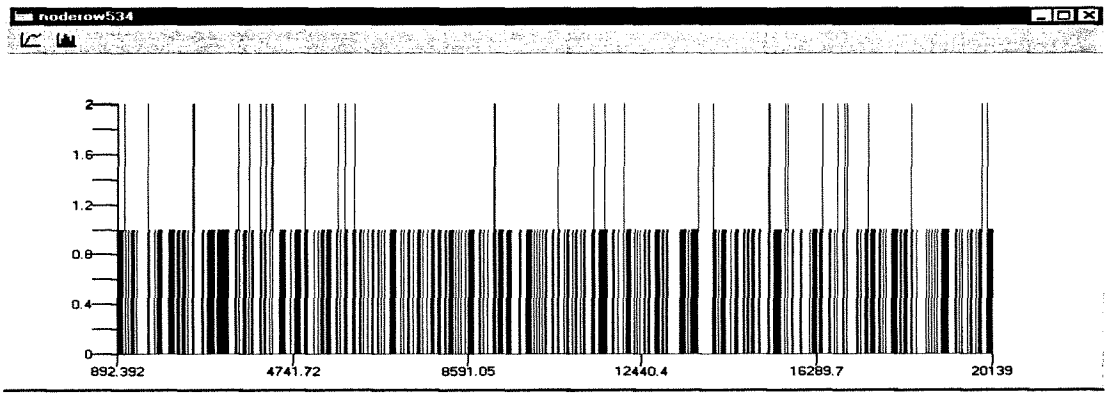


Figure 20.41 rgv queue at node 534

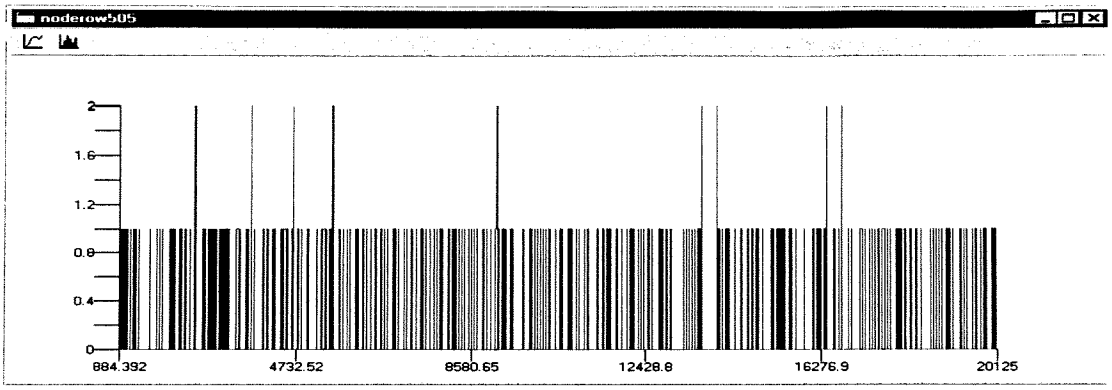


Figure 20.42 rgv queue at node 505

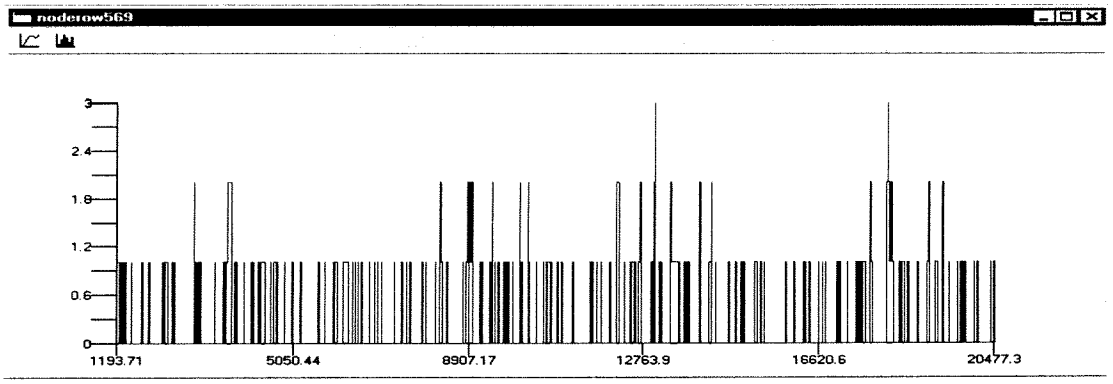


Figure 20.43 rgv queue at node 569

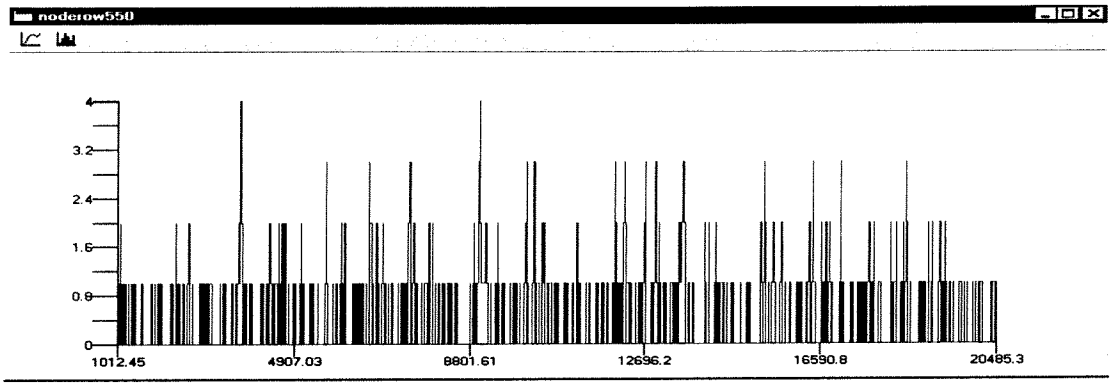


Figure 20.44 rgv queue at node 550

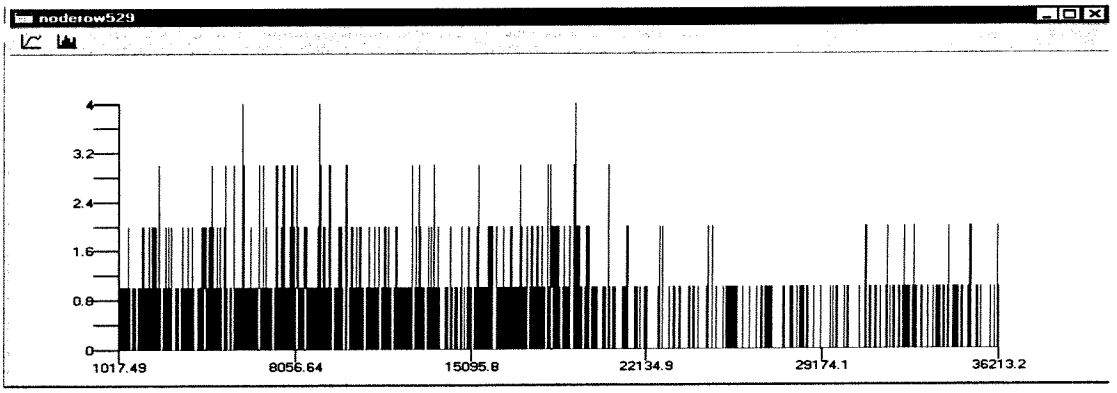


Figure 20.45 rgv queue at node 529

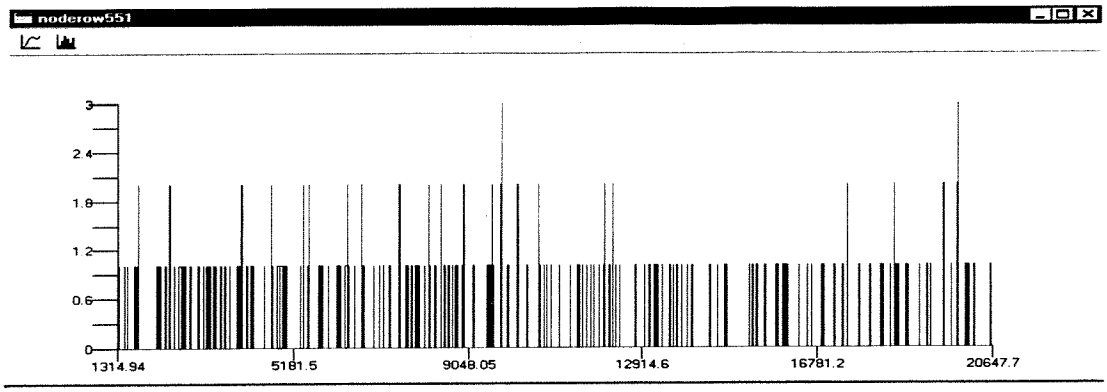


Figure 20.46 rgv queue at node 551

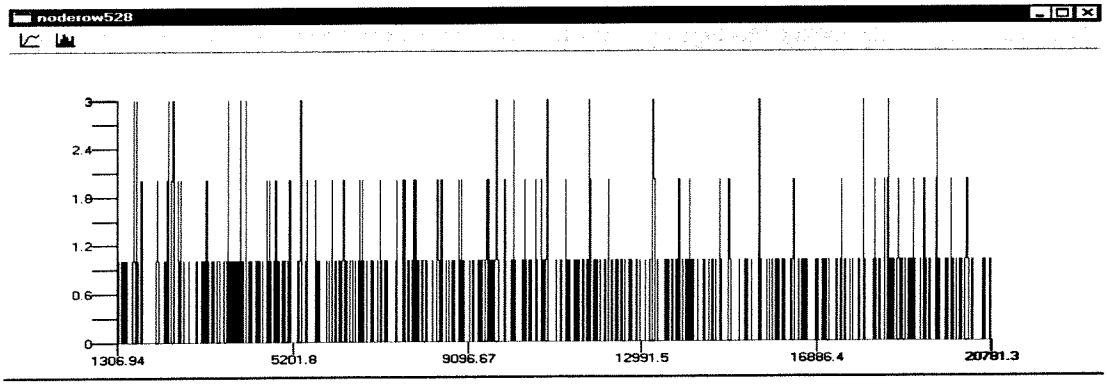


Figure 20.47 rgv queue at node 528

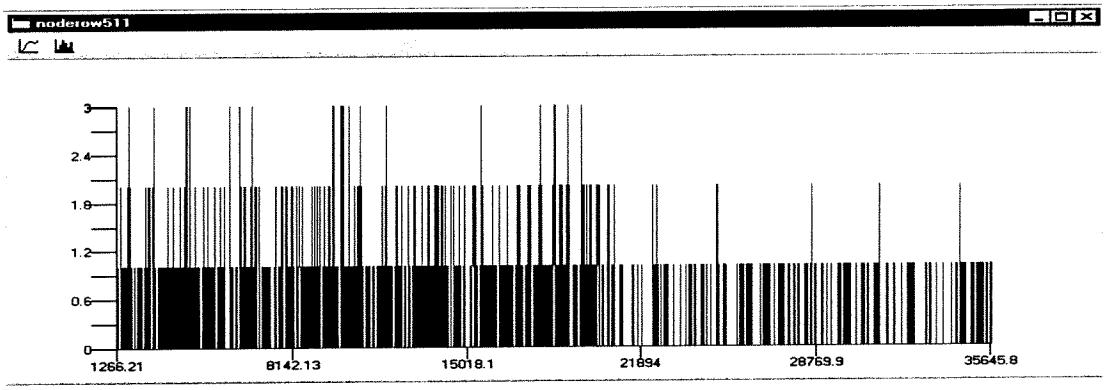


Figure 20.48 rgv queue at node 511

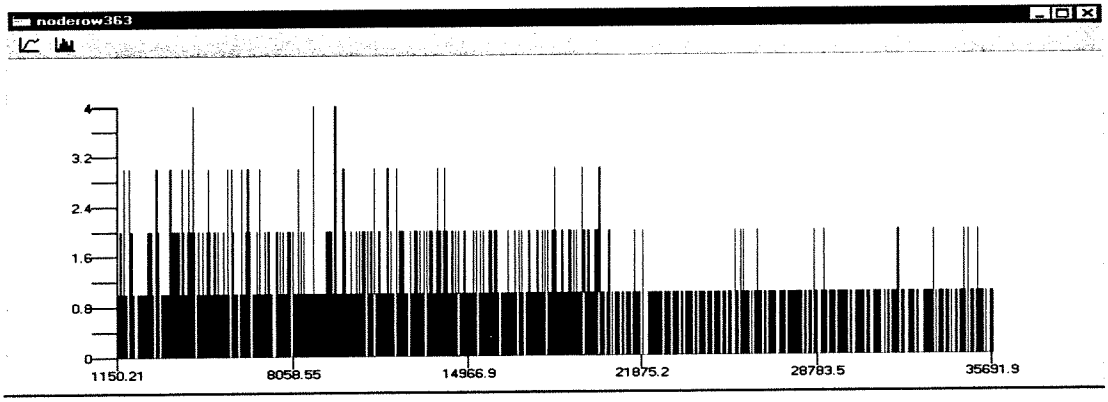


Figure 20.49 rgv queue at node 363

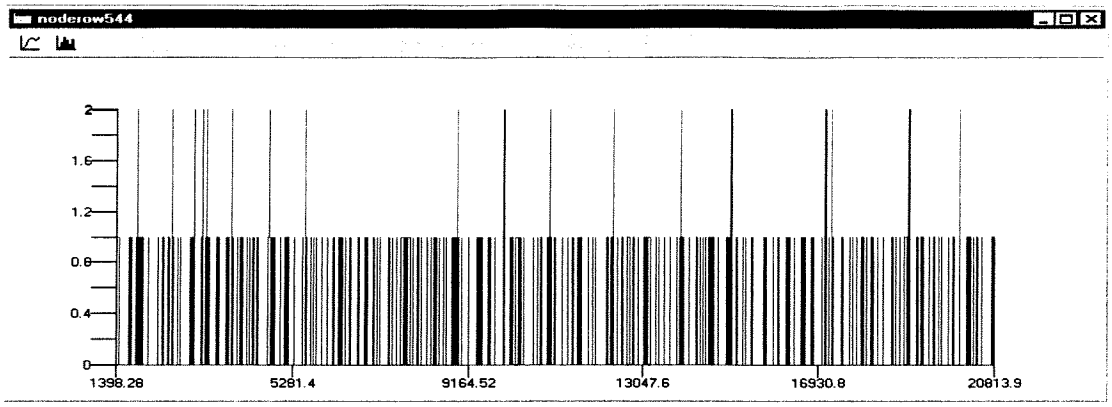


Figure 20.50 rgv queue at node 544

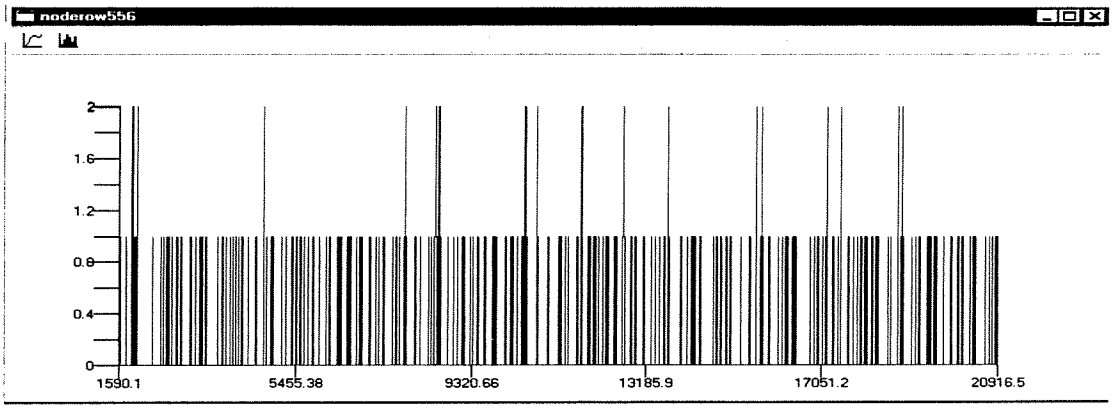


Figure 20.51 rgv queue at node 556

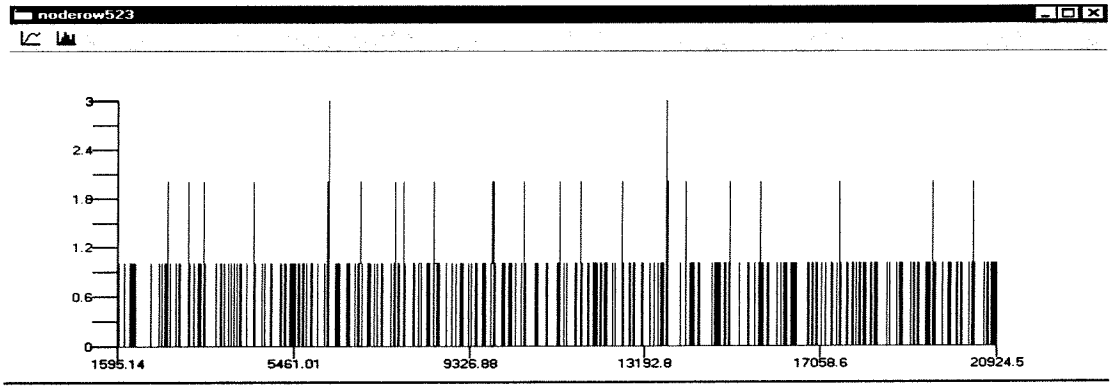


Figure 20.52 rgv queue at node 523

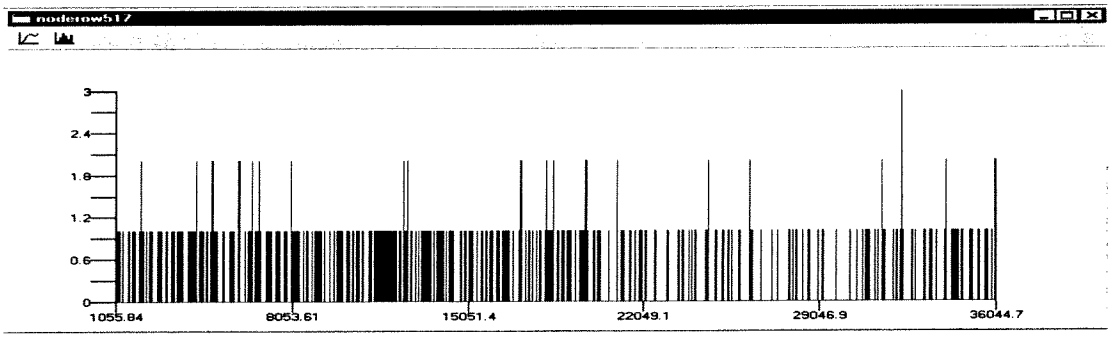


Figure 20.53 rgv queue at node 517

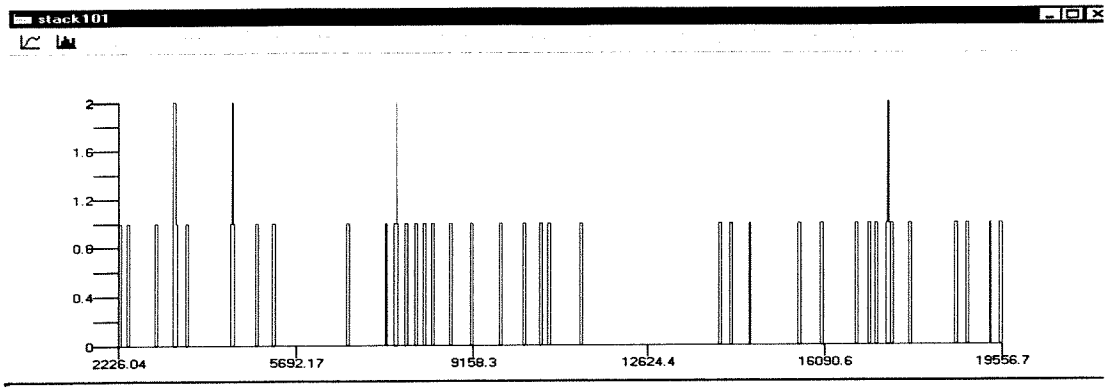


Figure 20.54 rgv queue in warehouse lane 101

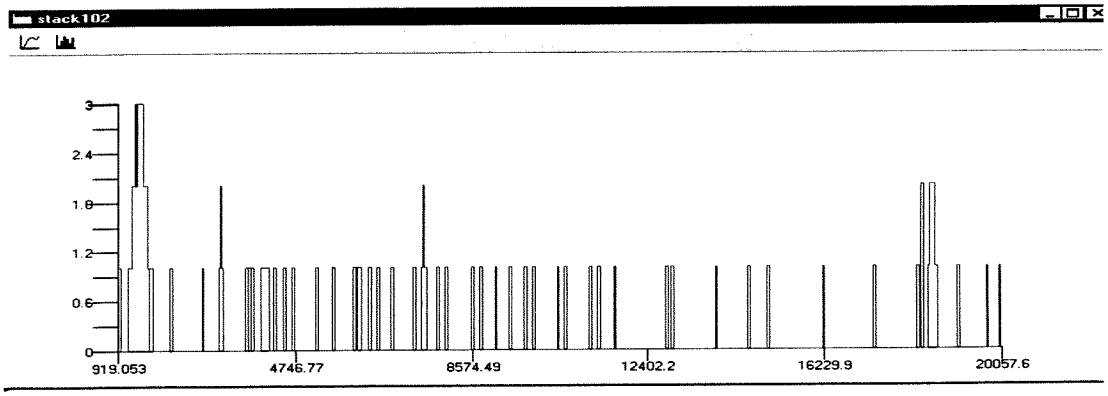


Figure 20.55 queue in warehouse lane 102

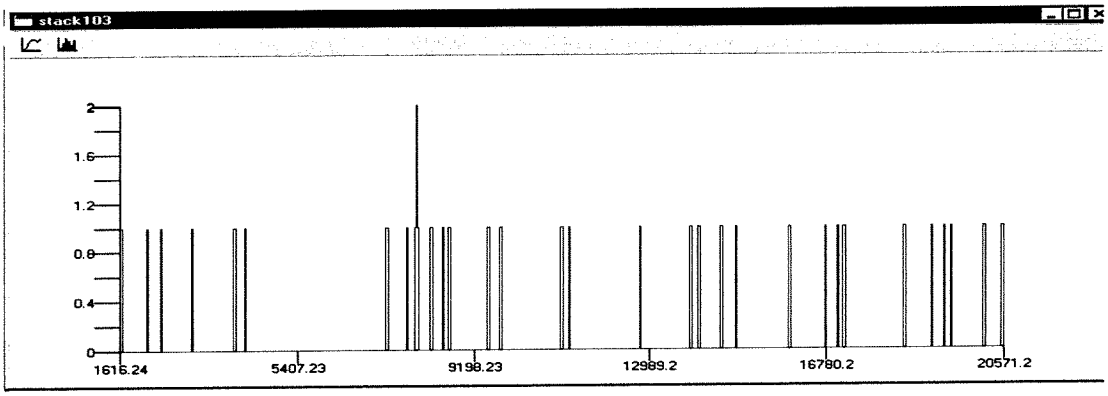


Figure 20.56 queue in warehouse lane 103

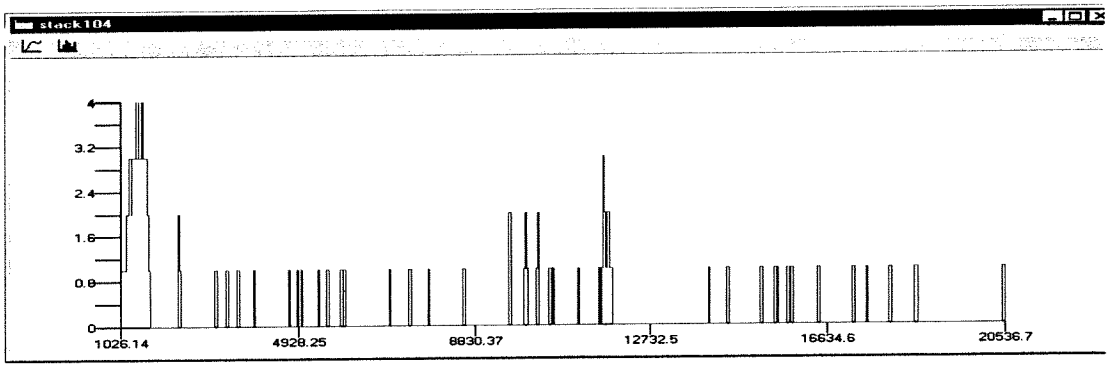


Figure 20.57 queue in warehouse lane 104

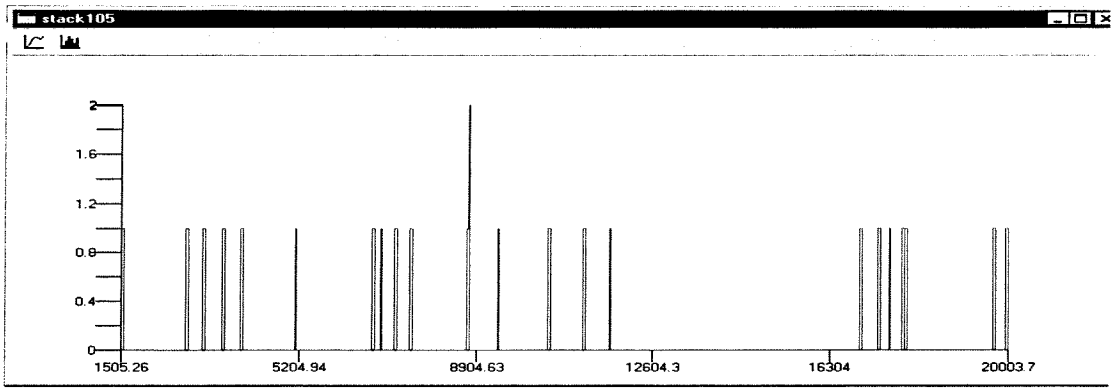


Figure 20.58 queue in warehouse lane 105

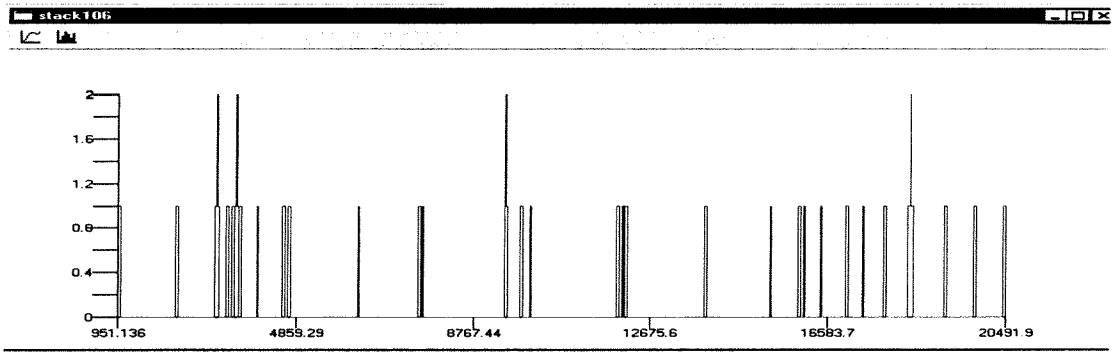


Figure 20.59 queue in warehouse lane 106

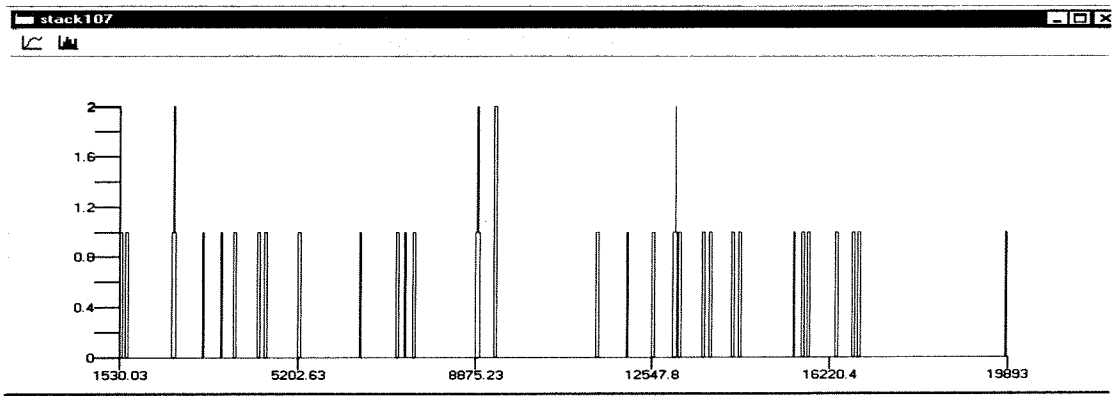


Figure 20.60 queue in warehouse lane 107

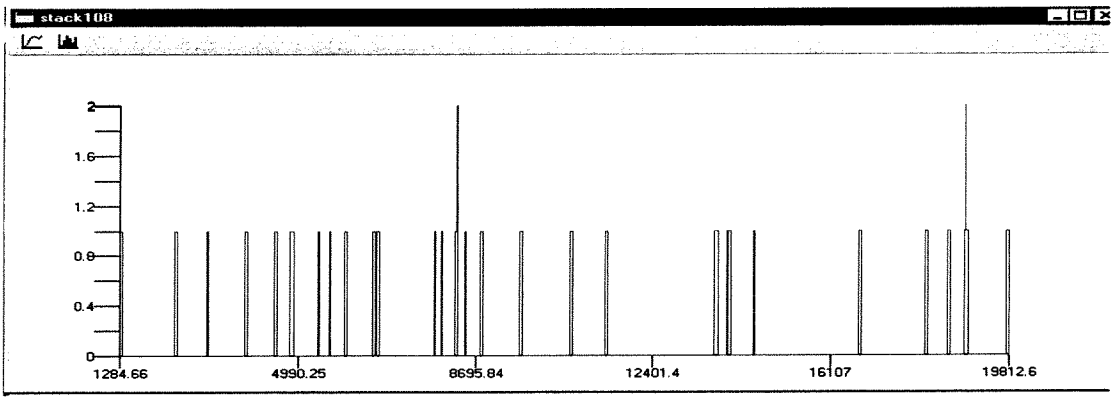


Figure 20.61 queue in warehouse lane 108

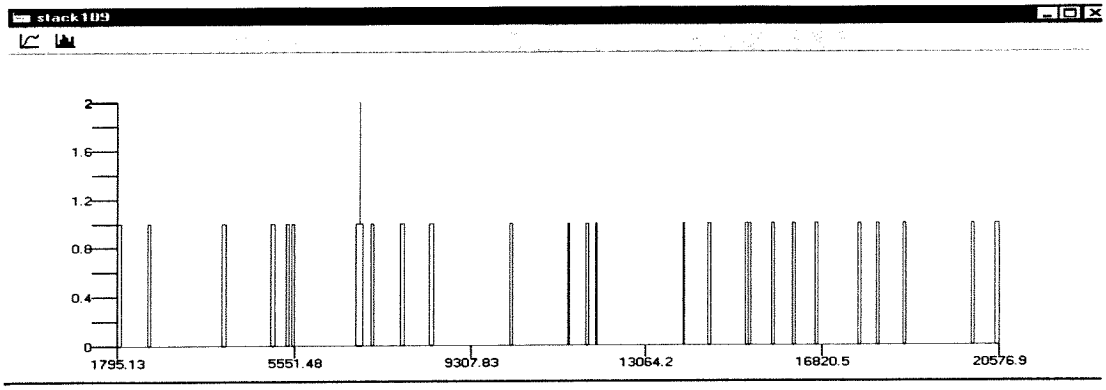


Figure 20.62 queue in warehouse lane 109

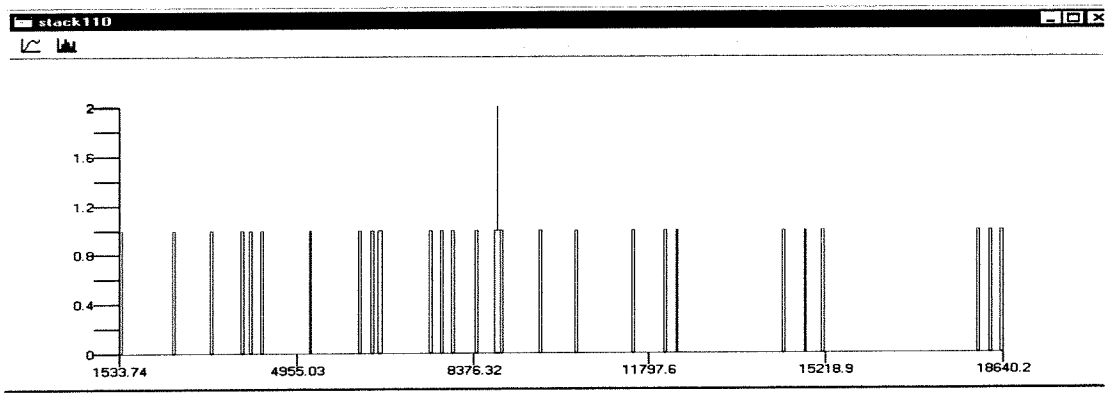


Figure 20.63 queue in warehouse lane 110

Appendix 21: Input parameters for the experiment adjacent loading at Berth 1 and 3 and unloading at Berth 2.

The basic parameters used for this experiment are shown in Table 21.1. The parameters are the same as used in experiment 10, except for the number of rgv's used when loading a vessel from an adjacent berth. In this experiment that is the case for Berth 1 and Berth 3.

Parameter	Value	
frombtob[1,2]	0	containers
frombtob[3,2]	0	containers
frombtob[2,1]	1600	containers
frombtob[2,3]	1600	containers
inplacecounter 1	0	containers
inplacecounter 2	0	containers
inplacecounter 3	0	containers
seed of uniform	67	-
seed of arrivaltimes feeders	17	-
seed of arrivaltimes big	27	-
import generator feeders	40	containers
export generator feeders	1050	containers
import generator big vessels	3150	containers
export generator big vessels	0	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
rgv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	70	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	58	seconds

Table 21.1 input parameters for experiment; Adjacent loading at Berth 1 and Berth 3 and unloading at Berth 2.

Parameter	Value	
frombtob[1,2]	40	containers
frombtob[3,2]	40	containers
frombtob[2,1]	2083	containers
frombtob[2,3]	2087	containers

Table 21.2 output parameter values for experiment; Adjacent loading at Berth 1 and Berth 3 and unloading at Berth 2.

21.1.1 Crane occupancy rates when adjacent loading on Berths 1&3 and unloading on Berth 2

As both feeder vessels will be provided from Yard 2 this means that 1050 * 2 = 2100 containers have to retrieved from these stacks. Both feeder vessels will leave the terminal after 5.3 hours. In this time span the mega vessel has unloaded 1960 containers into Yard 2. Adding these up to the 2100 containers loaded onto both feeder vessels will amount to 4060 containers.

Considering that the number of containers that will retrieved from the 20' stacks and the 40' stacks are the same means that 2030 containers will be retrieved from both stack types.

- Required 20' warehouse crane operational time: 2030 * 80 seconds = 162,400 seconds
- Required 40' warehouse crane operational time: 2030 * 80 seconds = 162,400 seconds

Number of warehouse cranes in Yard 2:
- Number of warehouse cranes in the 20' stacks in Yard 2 = 6 * 2 = 12 warehouse cranes
- Number of warehouse cranes in the 40' stacks in Yard 2 = 6 * 3 = 18 warehouse cranes

Needed operational time per warehouse crane:

- $162{,}400/12 = 13{,}533$ seconds per 20' stack warehouse crane
- $162{,}400/18 = 9{,}022$ seconds per 40' stack warehouse crane

Required adjacent loading time for both feeder vessels: 5.3 hours $\Rightarrow$ 19,080 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(13{,}533/19{,}080\right) * 100\% = 70.1\%$

40' stack: $\left(9{,}022/19{,}080\right) * 100\% = 47.3\%$

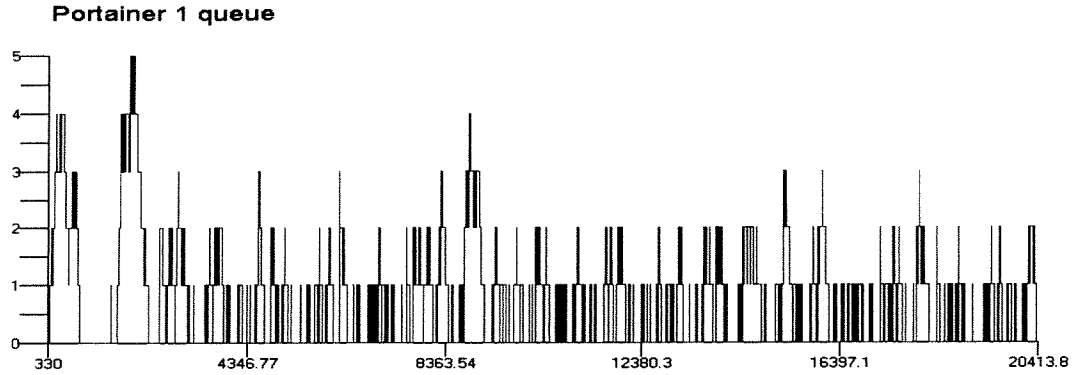


Figure 21.1, Portainer 1 queue at Berth 1 when adjacent loading a feeder

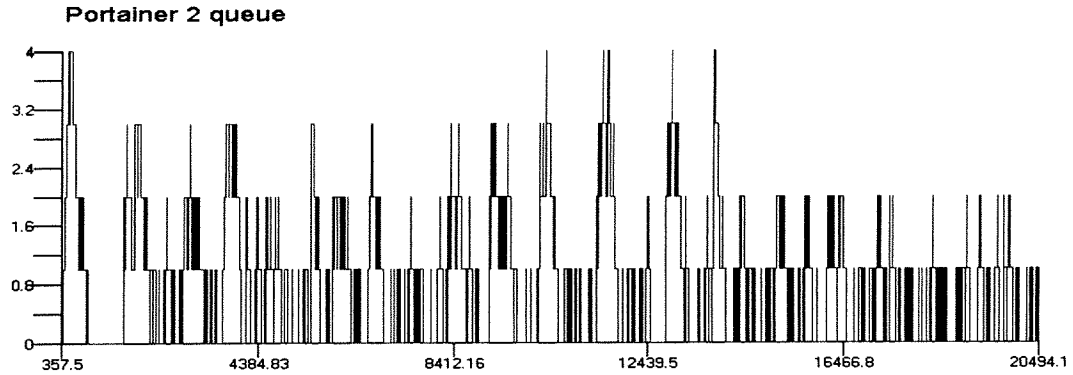


Figure 21.2 Portainer 2 queue at Berth 1 when adjacent loading a feeder

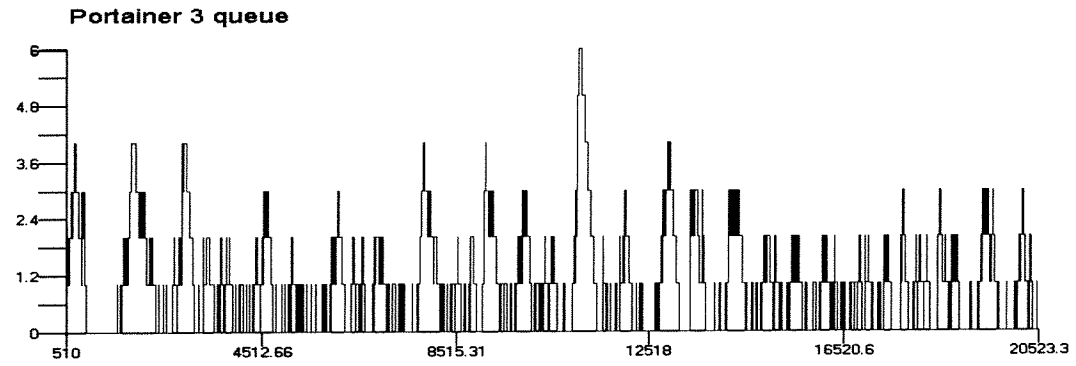


Figure 21.3 Portainer 3 queue at Berth 1 when adjacent loading a feeder

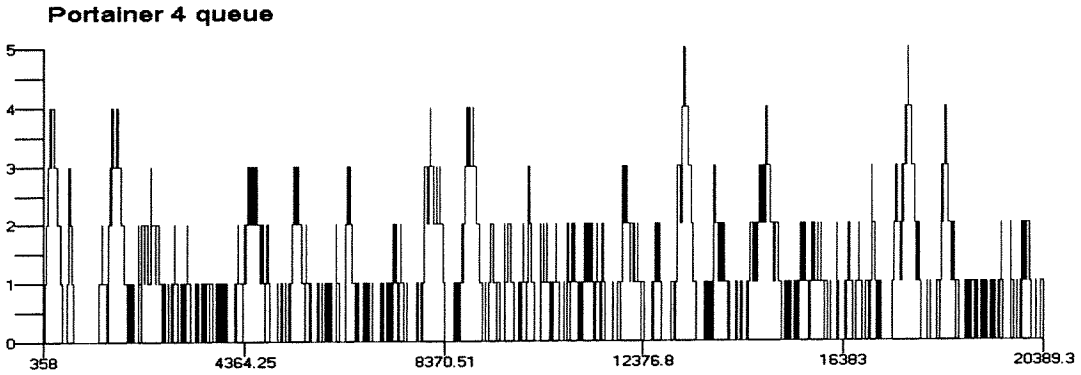


Figure 21.4 Portainer 4 queue at Berth 1 when adjacent loading a feeder

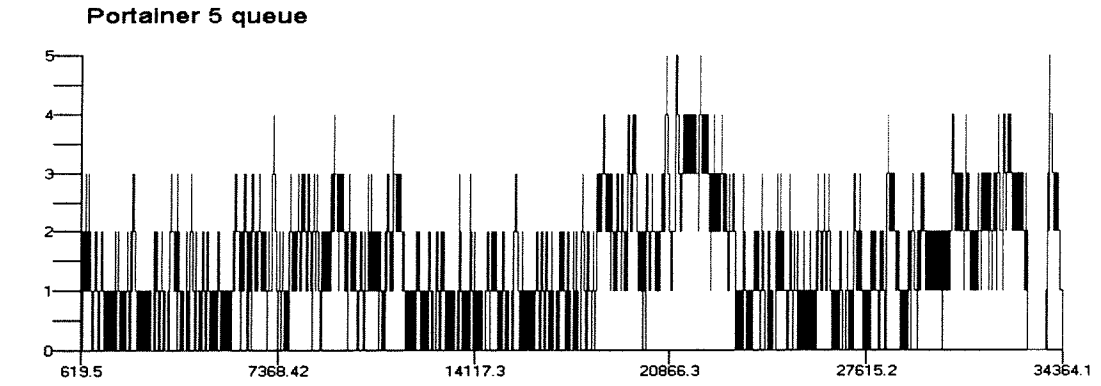


Figure 21.5 Portainer 5 queue at Berth 2 when unloading a mega vessel

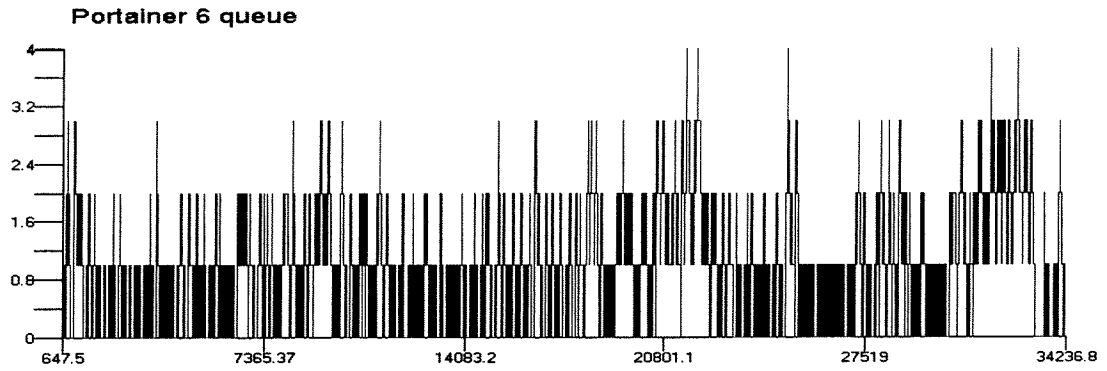


Figure 21.6 Portainer 6 queue at Berth 2 when unloading a mega vessel

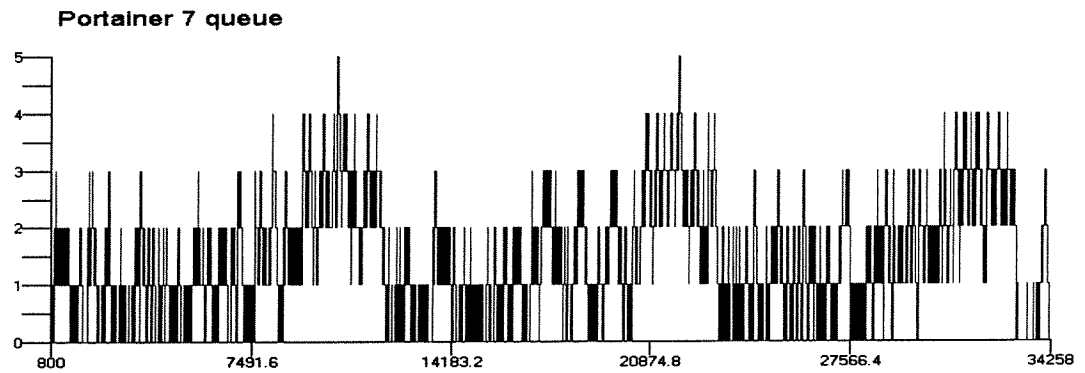


Figure 21.7 Portainer 7 queue at Berth 2 when unloading a mega vessel

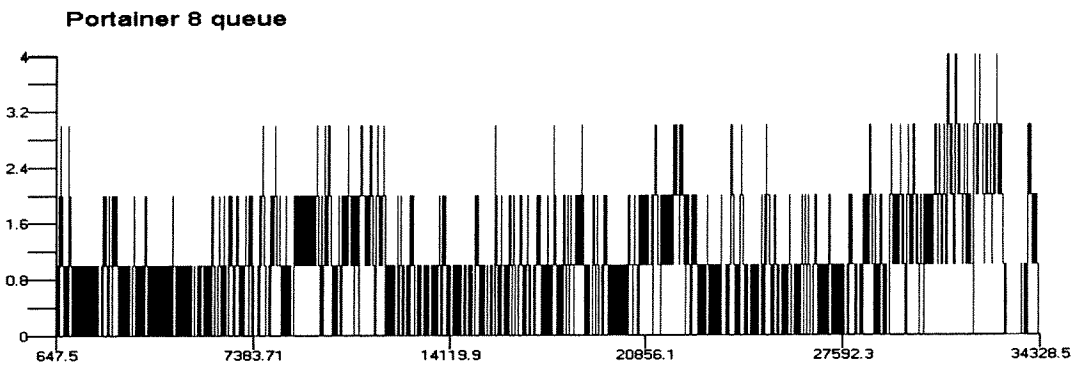


Figure 21.8 Portainer 8 queue at Berth 2 when unloading a mega vessel

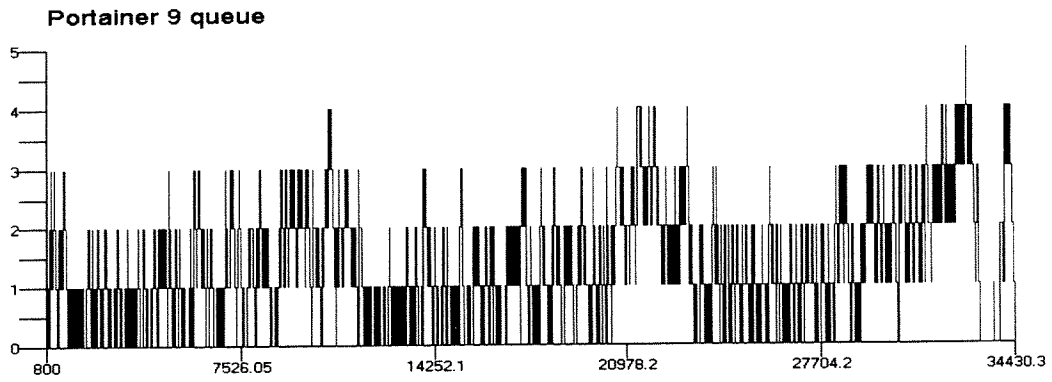


Figure 21.9 Portainer 9 queue at Berth 2 when unloading a mega vessel

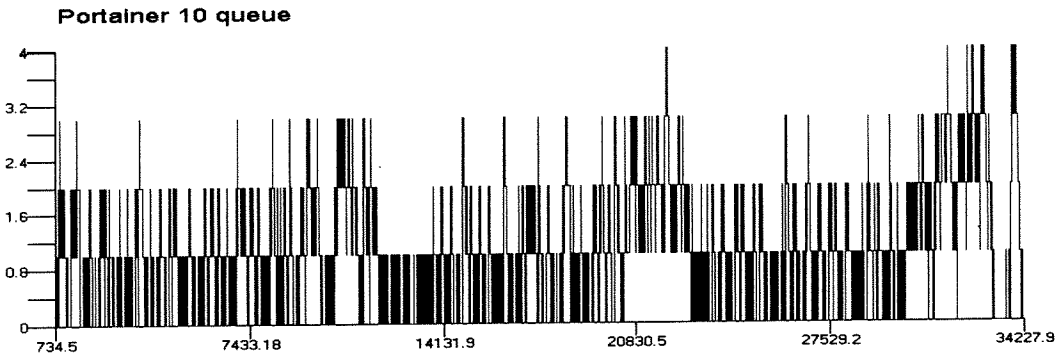


Figure 21.10 Portainer 10 queue at Berth 2 when unloading a mega vessel

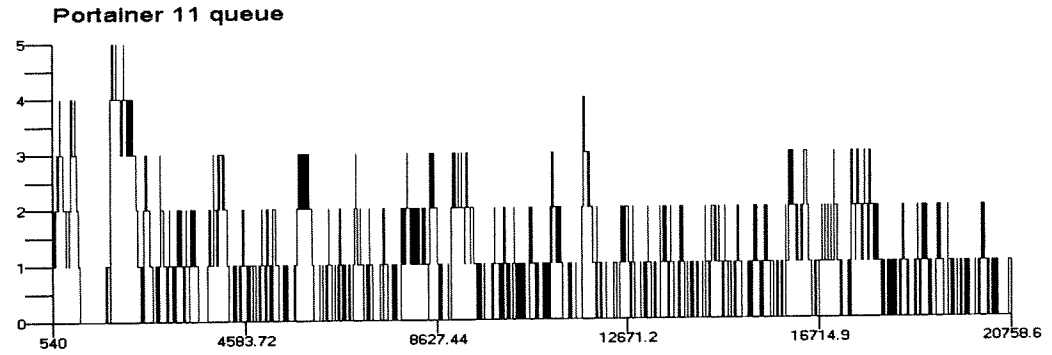


Figure 21.11 Portainer 11 queue at Berth 3 when adjacent loading a feeder

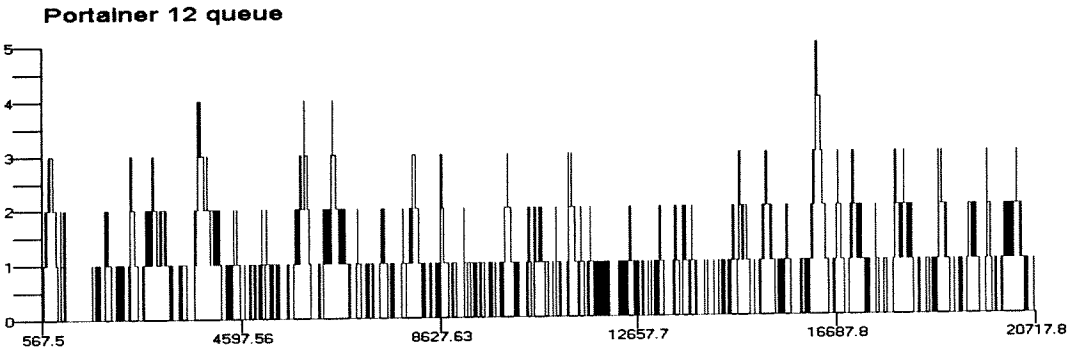


Figure 21.12 Portainer 12 queue at Berth 3 when adjacent loading a feeder

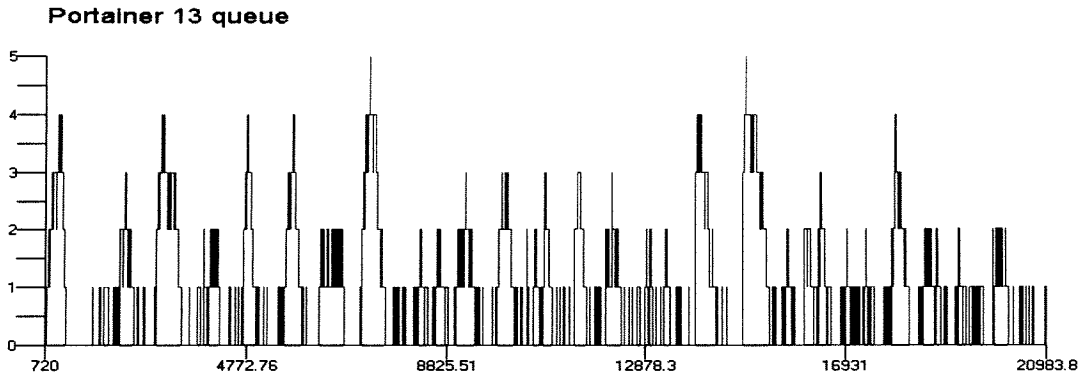


Figure 21.13 Portainer 13 queue at Berth 3 when adjacent loading a feeder

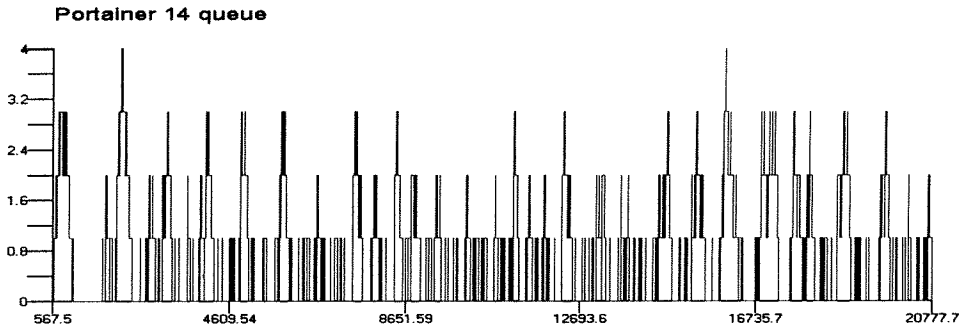


Figure 21.14 Portainer 14 queue at Berth 3 when adjacent loading a feeder

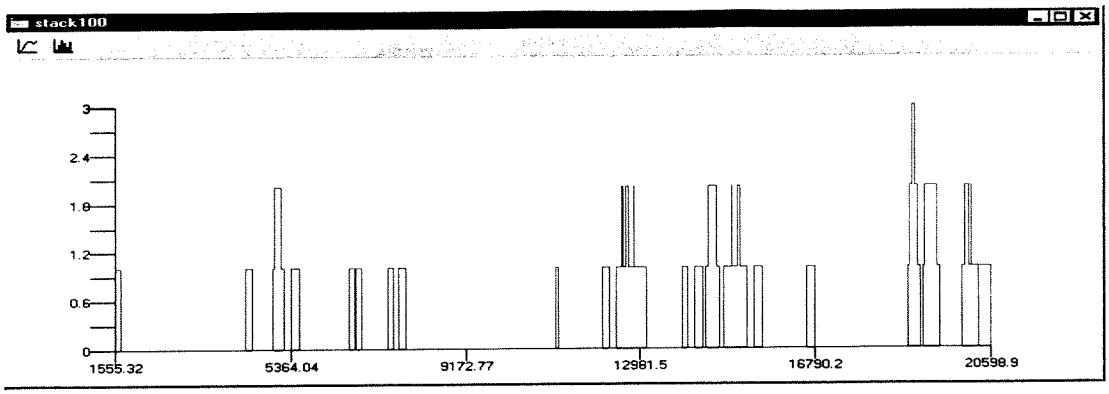


Figure 21.15 rgv queue on warehouse lane 100

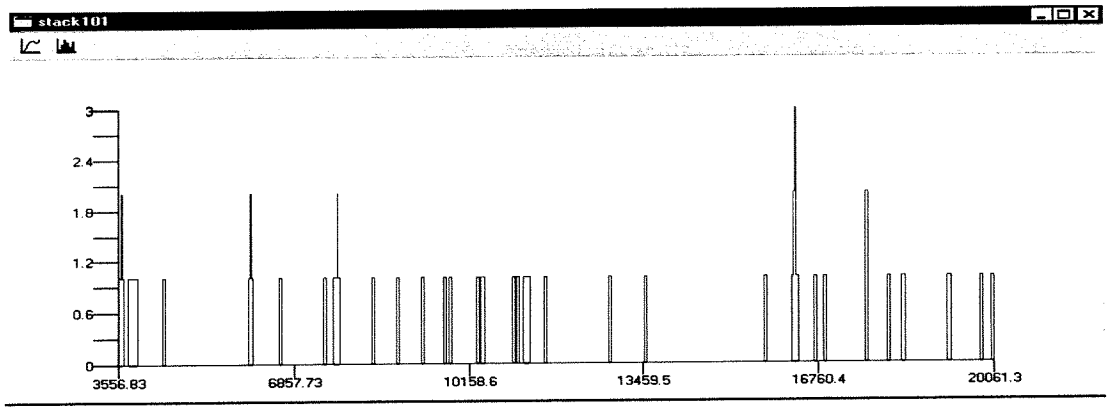


Figure 21.16 rgv queue on warehouse lane 101

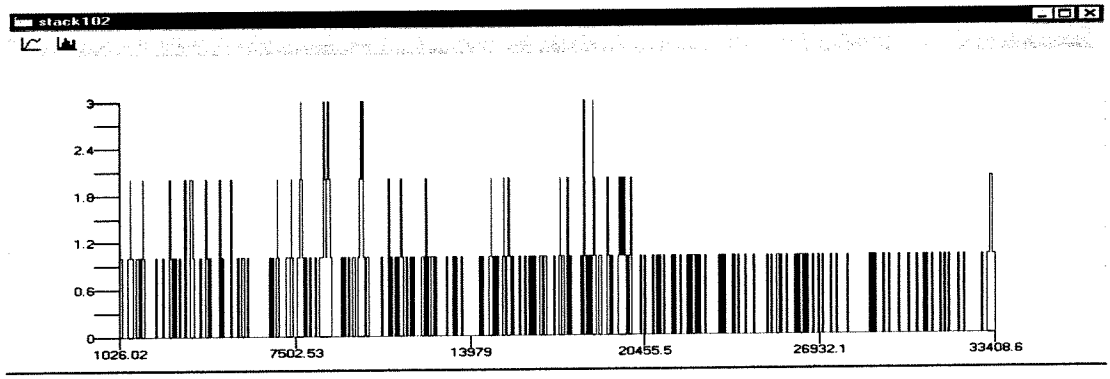


Figure 21.17 rgv queue on warehouse lane 102

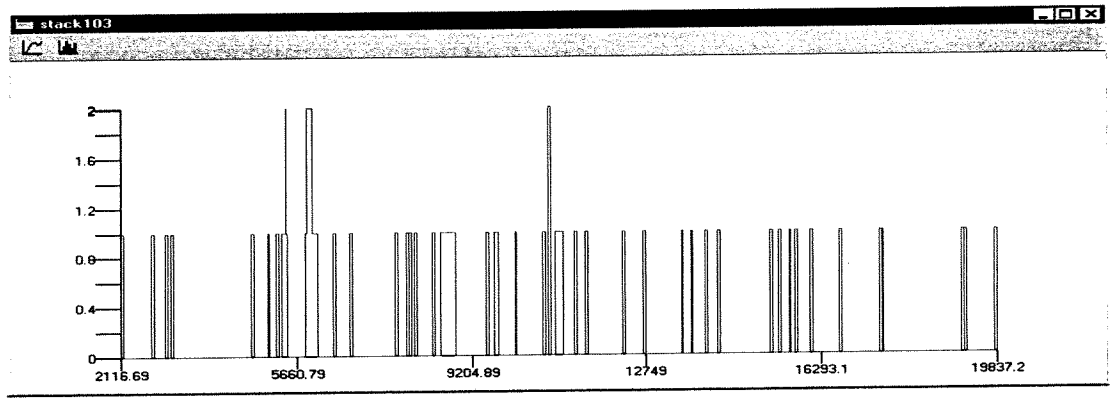


Figure 21.18 rgv queue on warehouse lane 103

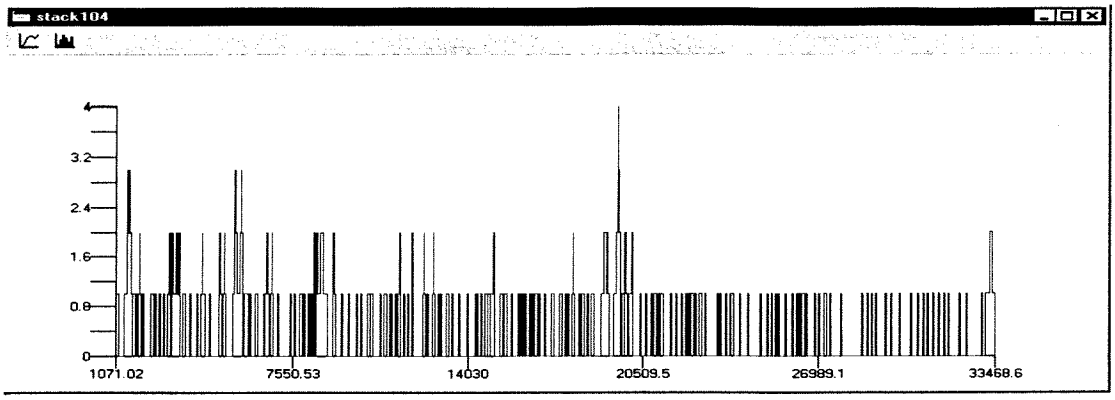


Figure 21.19 rgv queue on warehouse lane 104

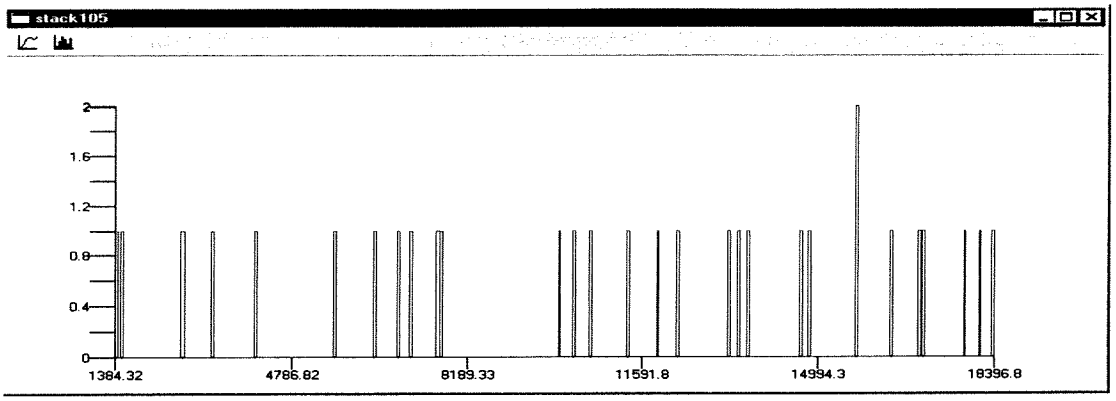


Figure 21.20 rgv queue on warehouse lane 105

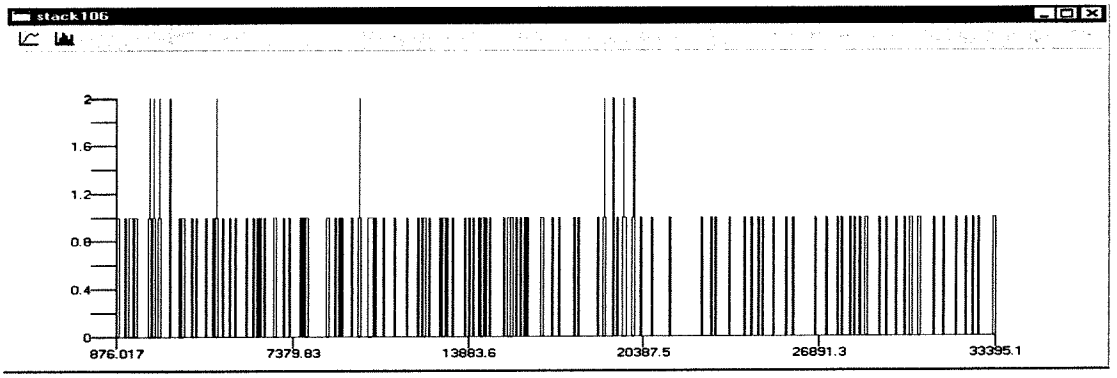


Figure 21.21 rgv queue on warehouse lane 106

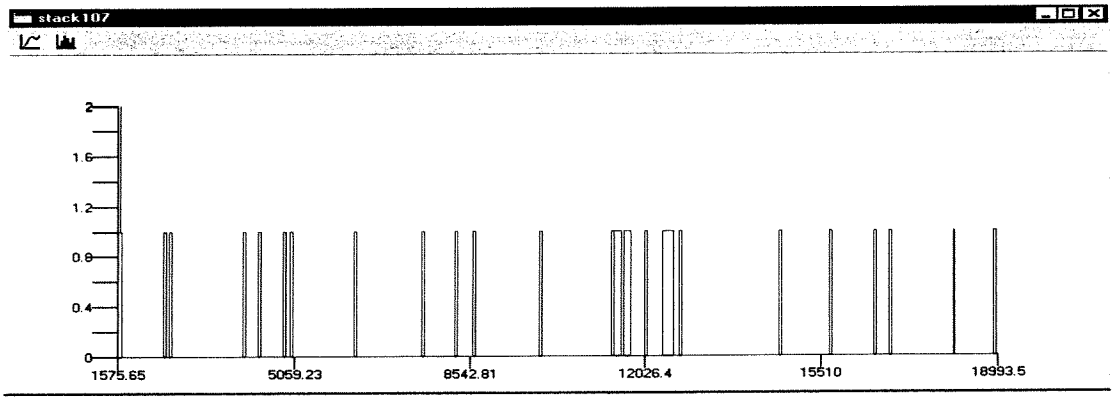


Figure 21.22 rgv queue on warehouse lane 107

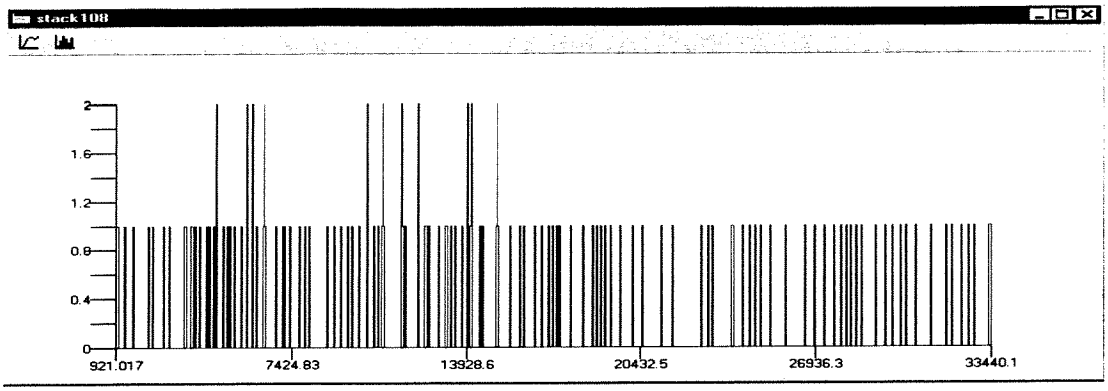


Figure 21.23 rgv queue on warehouse lane 108

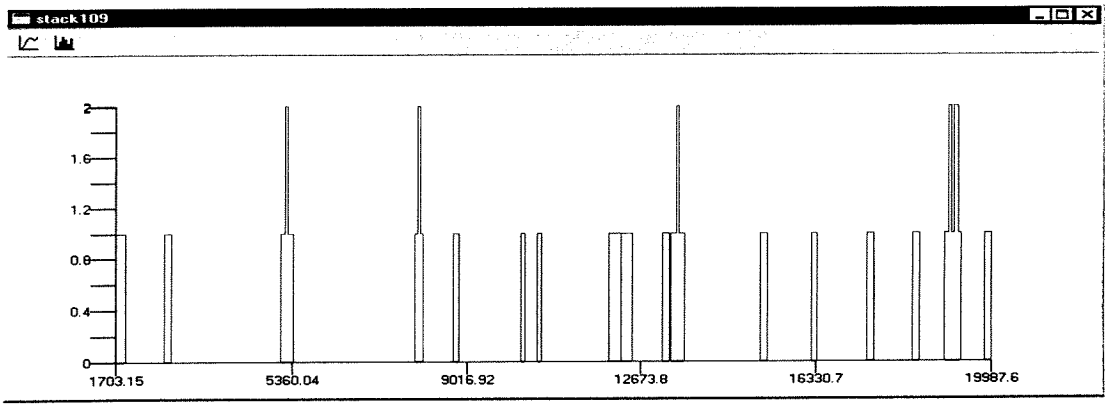


Figure 21.24 rgv queue on warehouse lane 109

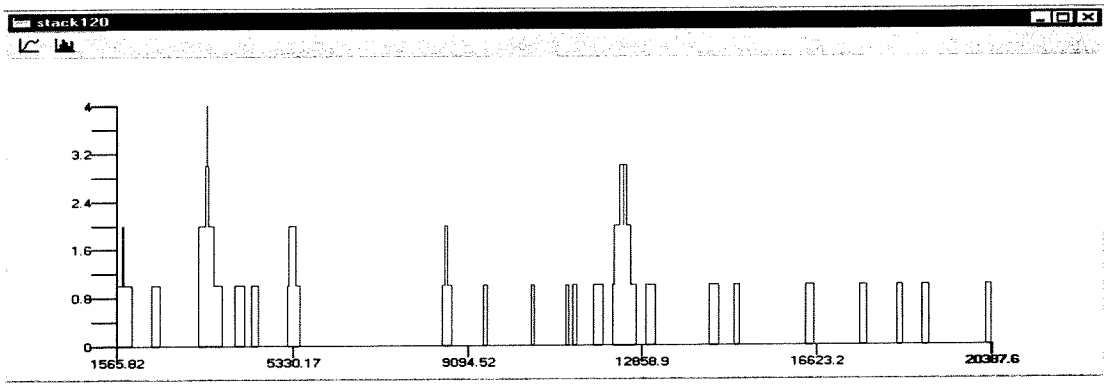


Figure 21.25 rgv queue on warehouse lane 120

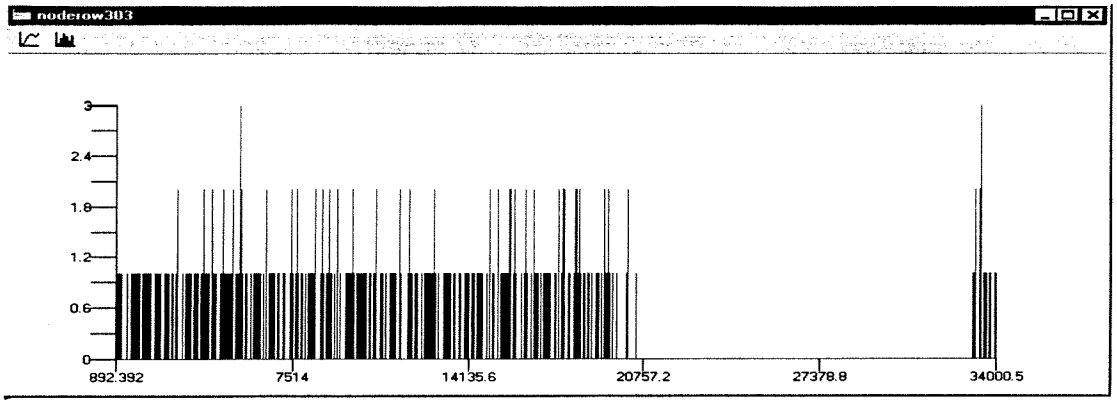


Figure 21.26 node row at node 303

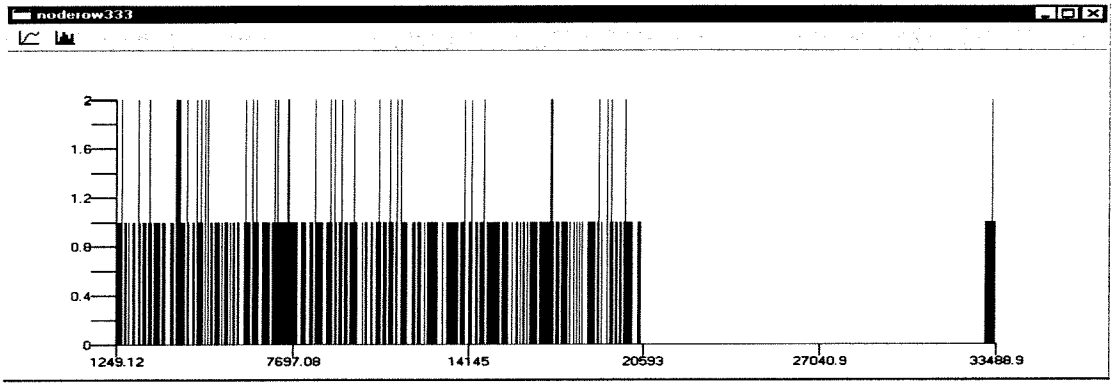


Figure 21.27 node row at node 333

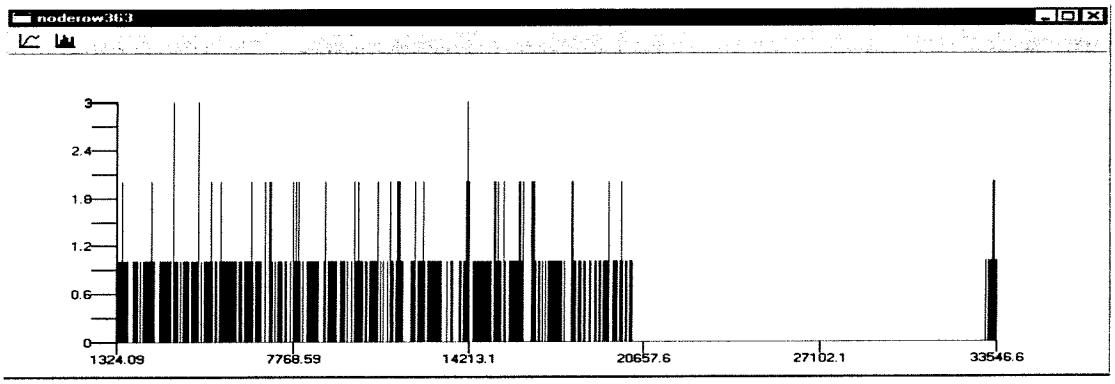


Figure 21.28 node row at node 363

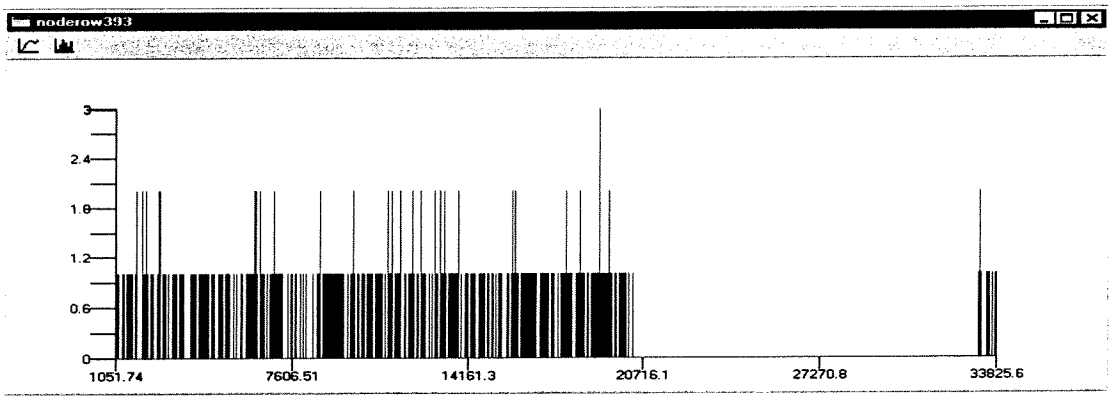


Figure 21.29 node row at node 393

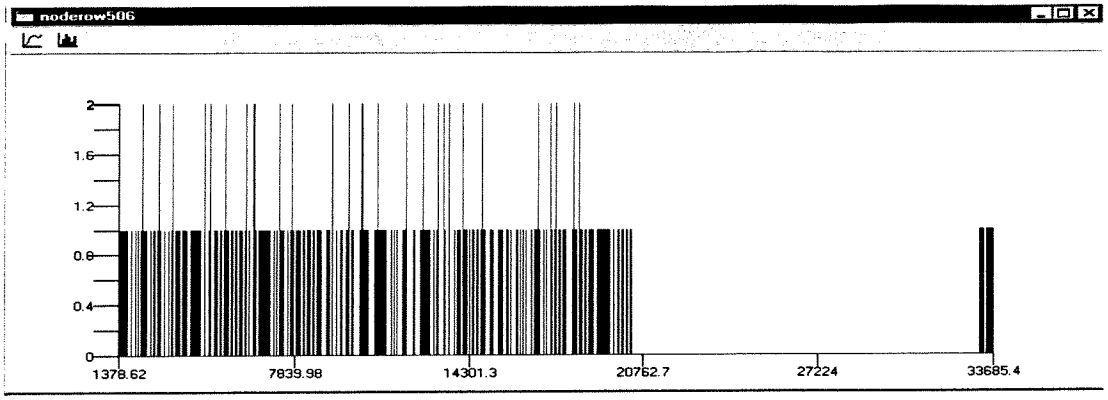


Figure 21.30 node row at node 506

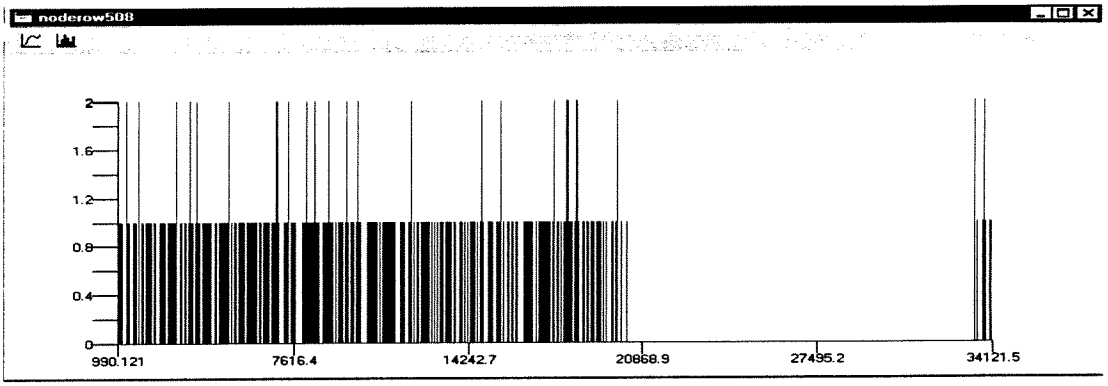


Figure 21.31 node row at node 508

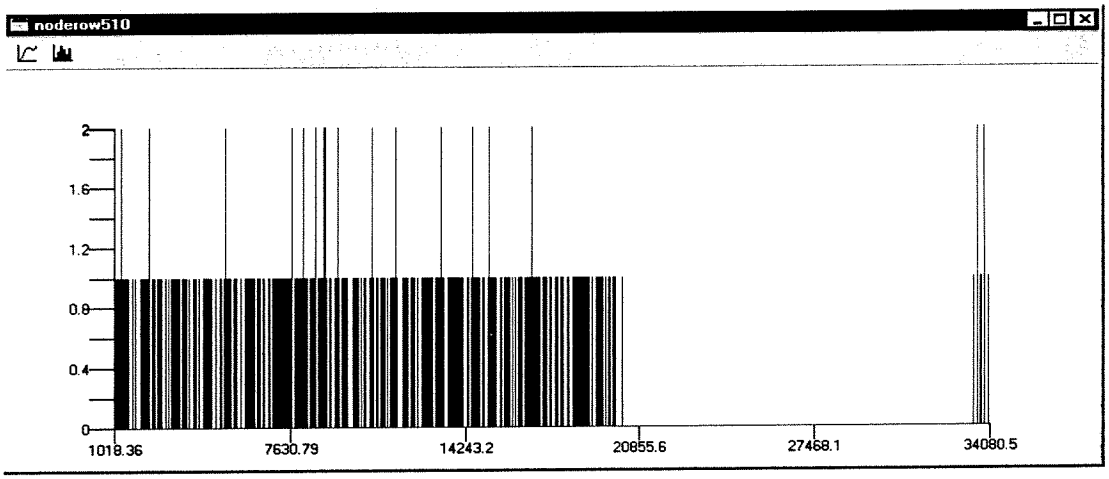


Figure 21.32 node row at node 510

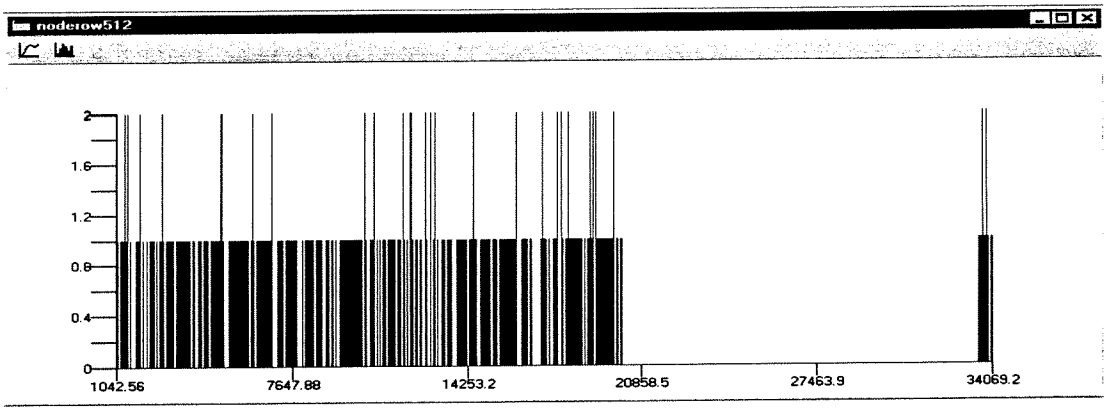


Figure 21.33 node row at node 512

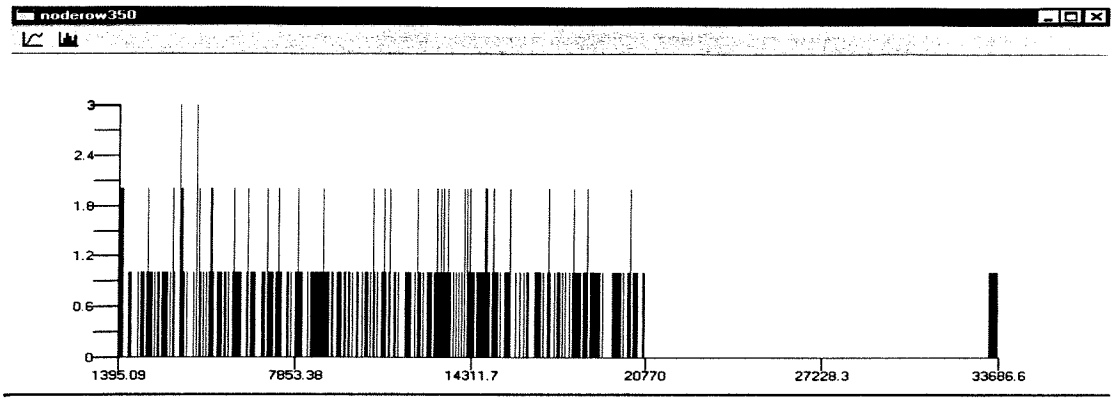


Figure 21.34 node row at node 350

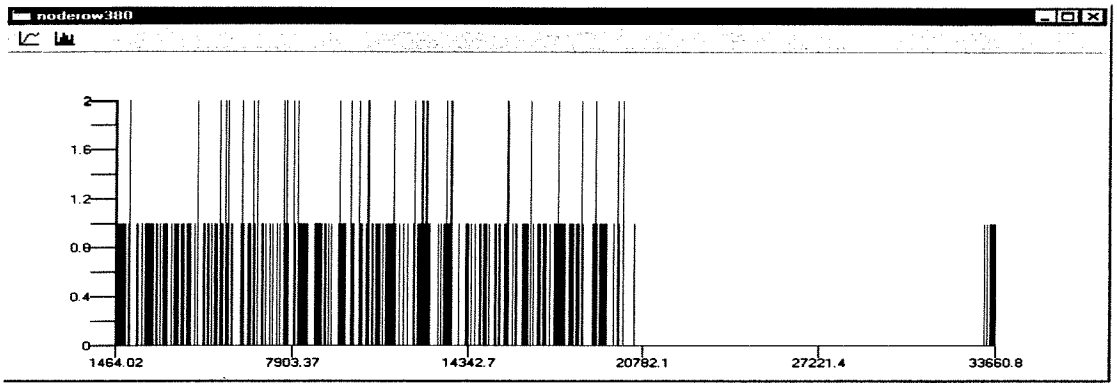


Figure 21.35 node row at node 380

Appendix 22: Input parameters for the experiment adjacent loading on Berth 2, unloading on Berths 1 & 3

The basic parameters used for this experiment are shown in Table 22.1. The parameter values after the run are shown in Table 22.2.

Parameter	Value	
frombtob[1,2]	2000	containers
frombtob[3,2]	2000	containers
frombtob[2,1]	0	containers
frombtob[2,3]	0	containers
inplacecounter 1	50	containers
inplacecounter 2	0	containers
inplacecounter 3	50	containers
seed of uniform	67	-
seed of arrivaltimes feeders	17	-
seed of arrivaltimes big	27	-
import generator feeders	1050	containers
export generator feeders	40	containers
import generator big vessels	60	containers
export generator big vessels	3150	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
agv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	70	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	58	seconds

Table 22.1: Input parameters for experiment, adjacent loading on Berth 2, unloading on Berth 1 & Berth 3

Parameter	Value	
frombtob[1,2]	1510	containers
frombtob[3,2]	1440	containers
frombtob[2,1]	34	containers
frombtob[2,3]	26	containers
Inplacecounter 1	10	containers
Inplacecounter 2	0	containers
Inplacecounter 3	10	containers

Table 22.2, output after experiment, adjacent loading on Berth 2, unloading on Berth 1

22.1.1 Situation 11: Adjacent loading on Berth 2; unloading on Berths 1 & 3

As the feeder vessels will unload in Yards 1 & 3 this means that 1050 containers have to be deposited in these stacks. The rgv's adjacent loading for Berth 2 will also retrieve their containers from Yards 1 & 3, bringing the total of handled containers to 3150/2 + 1050 = 2125.

Considering that the number of containers that will retrieved from the 20' stacks and the 40' stacks are the same means that 1063 containers will be retrieved from both stack types.

- Required 20' warehouse crane operational time: 1063 * 80 seconds = 85,040 seconds
- Required 40' warehouse crane operational time: 1063 * 80 seconds = 85,040 seconds

Number of warehouse cranes in Yard 2:
- Number of warehouse cranes in the 20' stacks in Yard 1 = 4 * 2 = 8 warehouse cranes
- Number of warehouse cranes in the 40' stacks in Yard 1 = 4 * 3 = 12 warehouse cranes

Needed operational time per warehouse crane:

- $85{,}040/8 = 10{,}630$ seconds per 20' stack warehouse crane
- $85{,}040/12 = 7{,}090$ seconds per 40' stack warehouse crane

Required adjacent loading time for both feeder vessels: 4.5 hours $\Rightarrow$ 16,200 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(10{,}630/16{,}200\right) * 100\% = 65.6\,\%$

40' stack: $\left(7{,}090/16{,}200\right) * 100\% = 43.7\,\%$

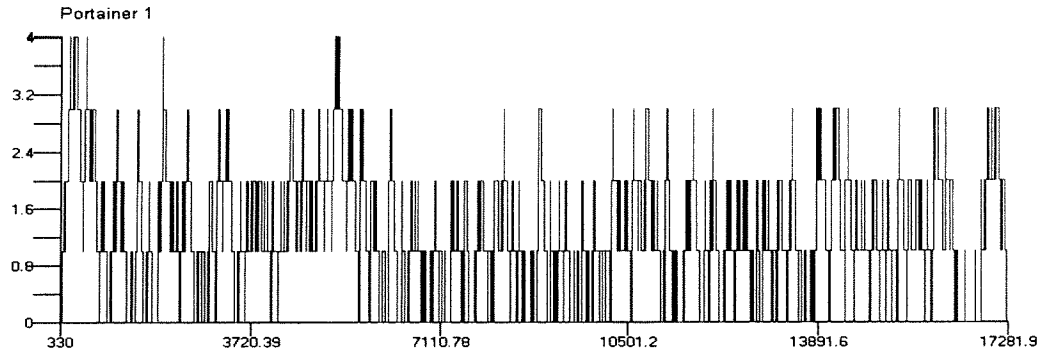


Figure 22.1, rgv row under portainer 1

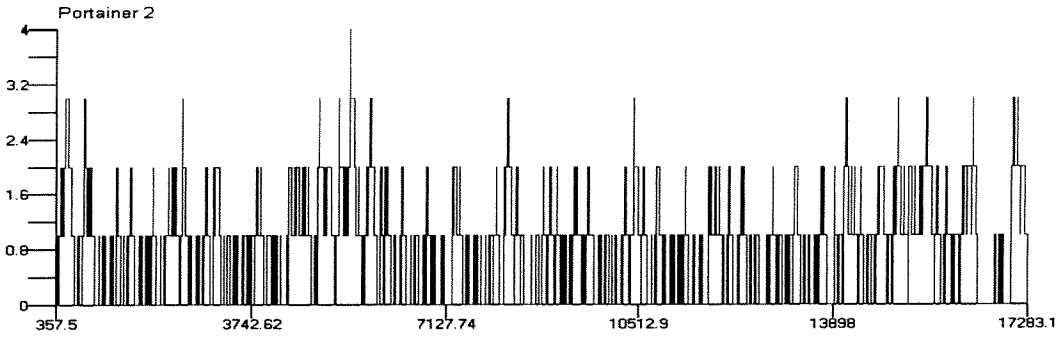


Figure 22.2, rgv row under portainer 2

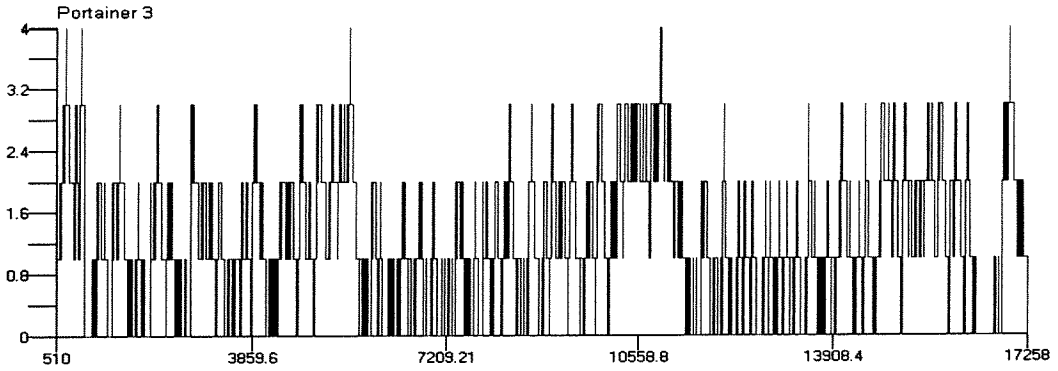


Figure 22.3, rgv row under portainer 3

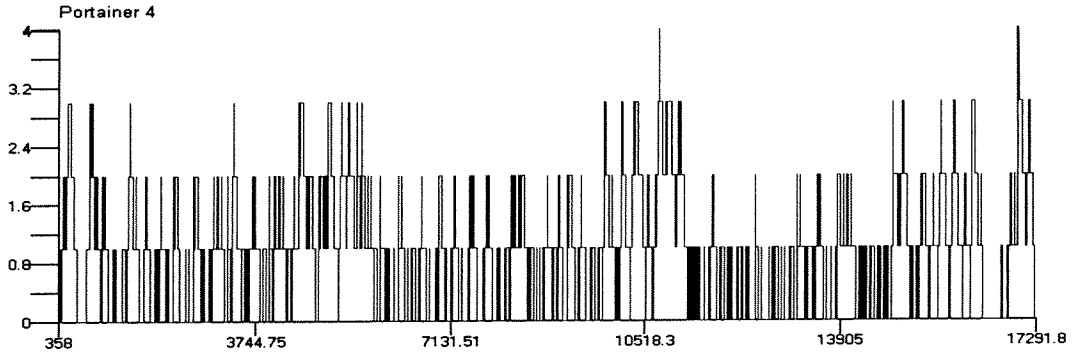


Figure 22.4, rgv row under portainer 4

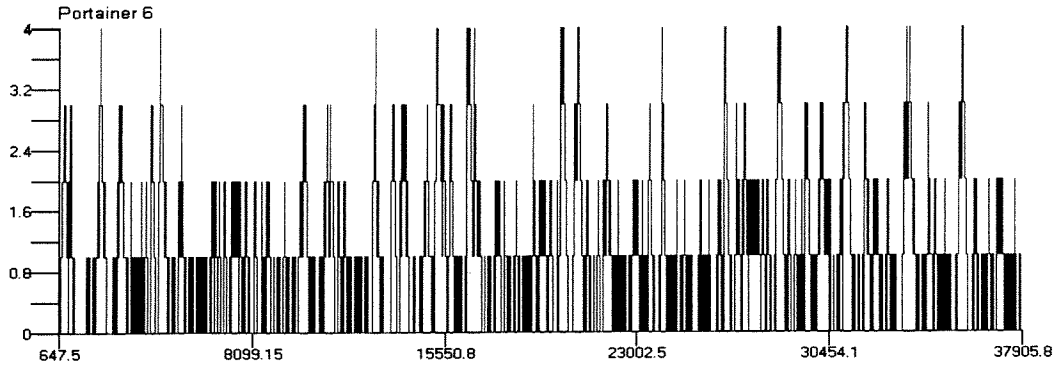


Figure 22.5, rgv row under portainer 6

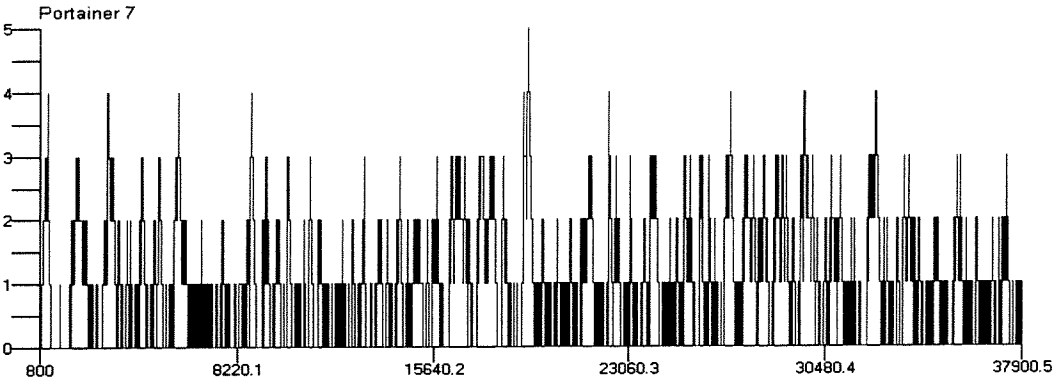


Figure 22.6, rgv row under portainer 7

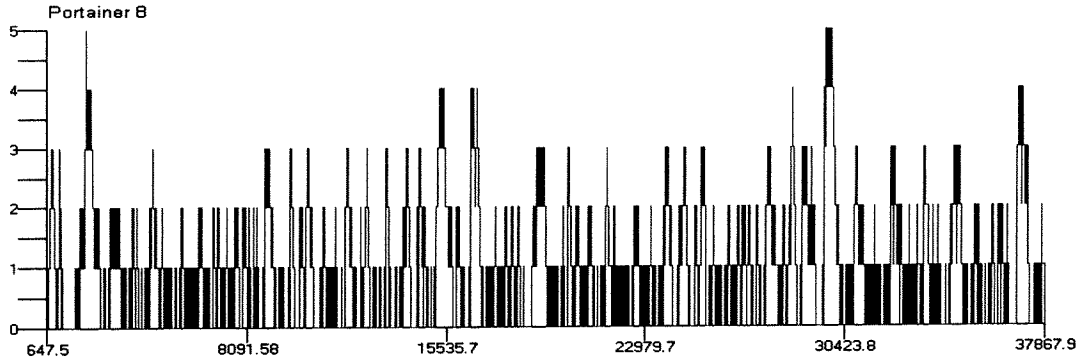


Figure 22.7, rgv row under portainer 8

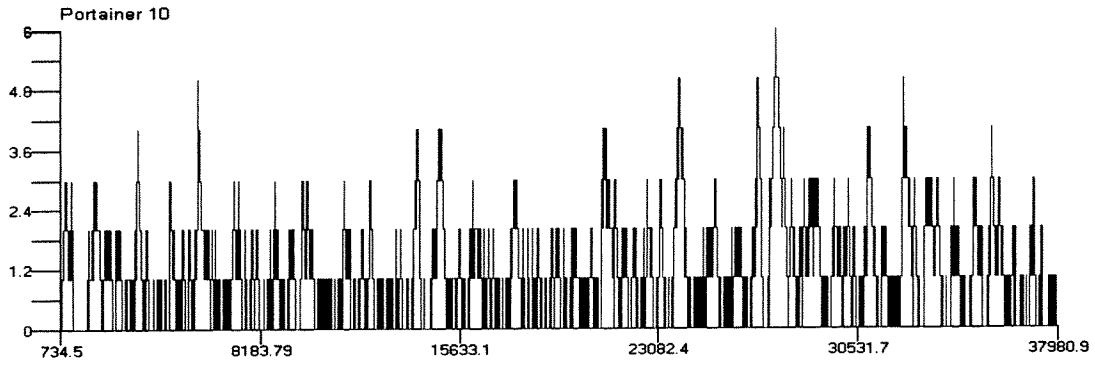


Figure 22.8, rgv row under portainer 10

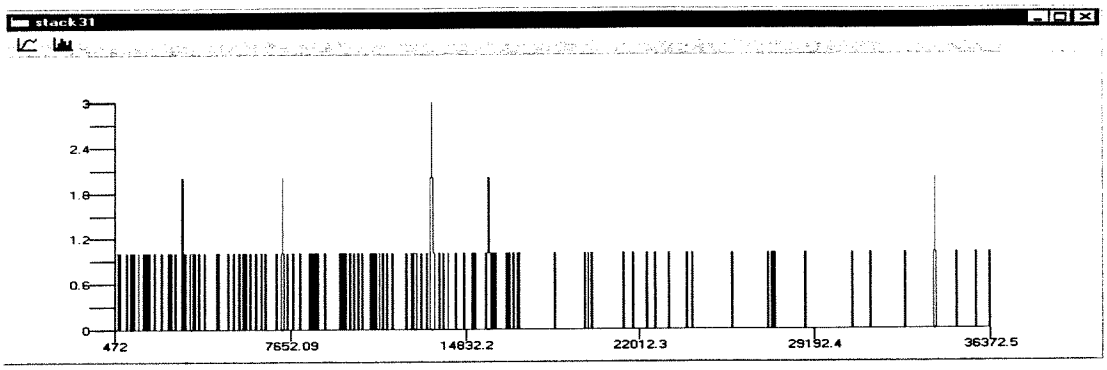


Figure 22.9, rgv row on warehouse lane 31

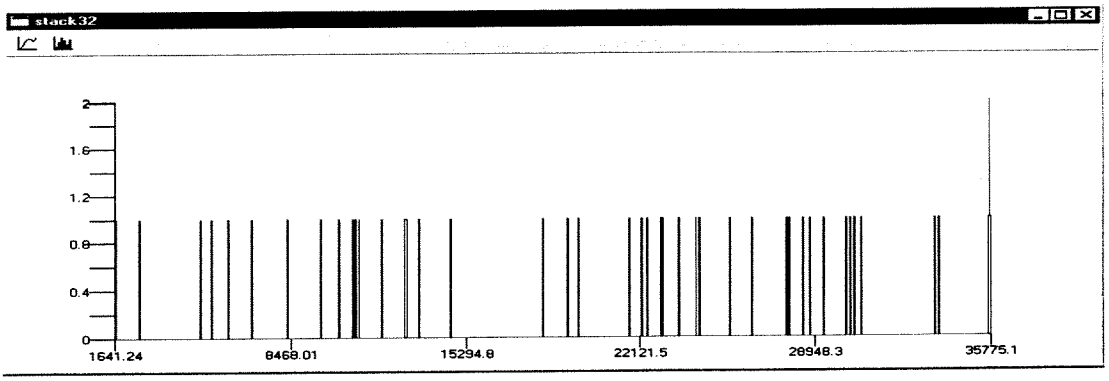


Figure 22.10, rgv row on warehouse lane 32

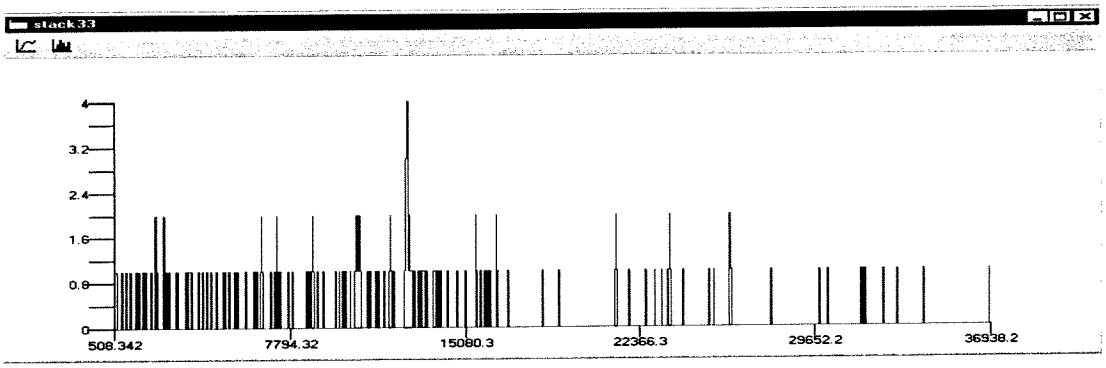


Figure 22.11, rgv row on warehouse lane 33

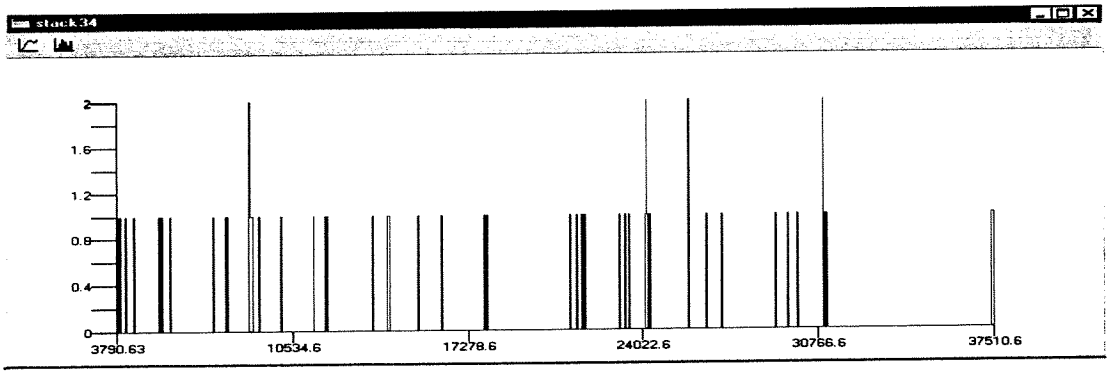


Figure 22.12, rgv row on warehouse lane 34

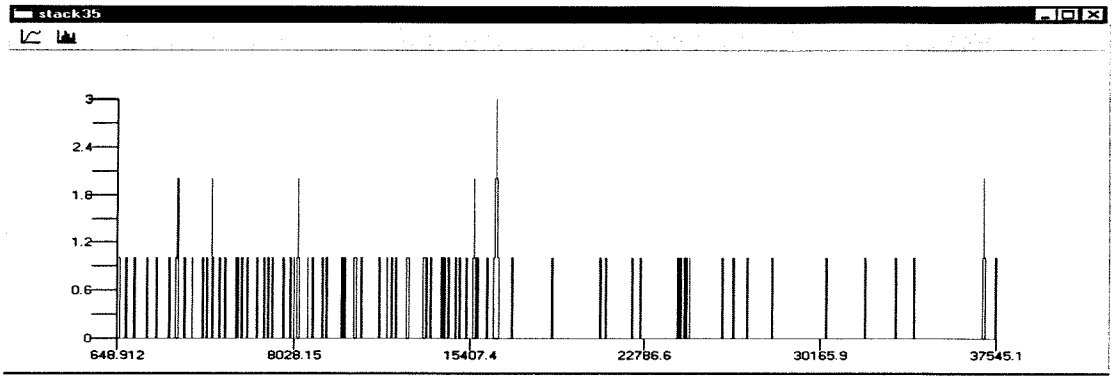


Figure 22.13, rgv row on warehouse lane 35

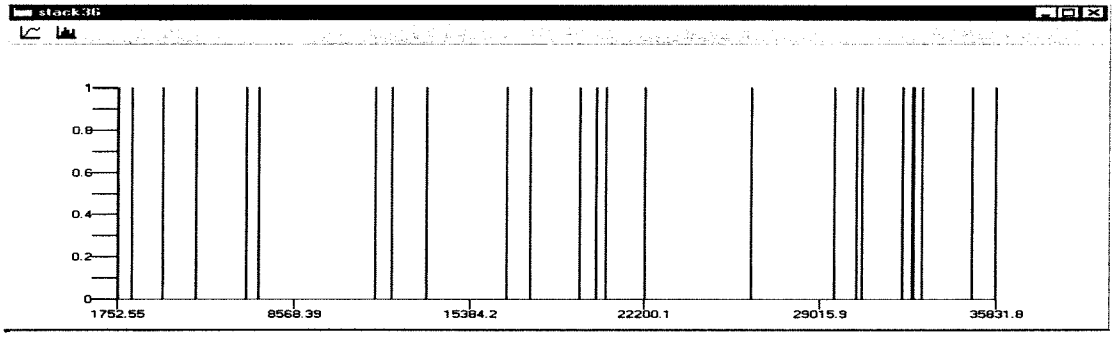


Figure 22.14, rgv row on warehouse lane 36

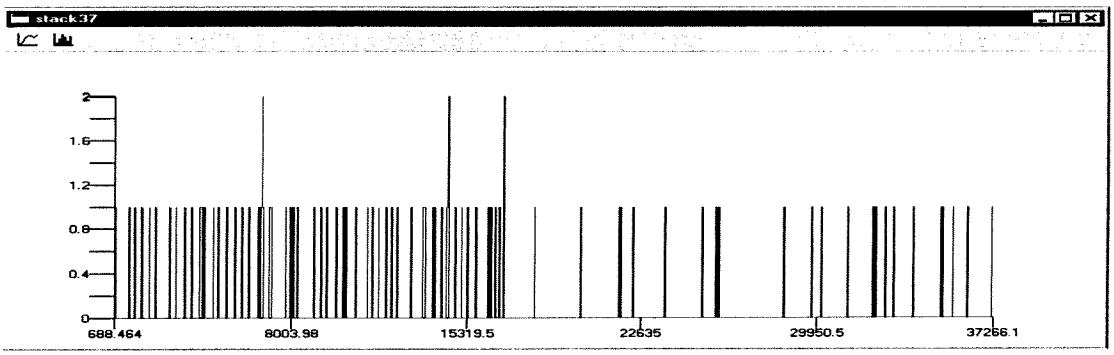


Figure 22.15, rgv row on warehouse lane 37

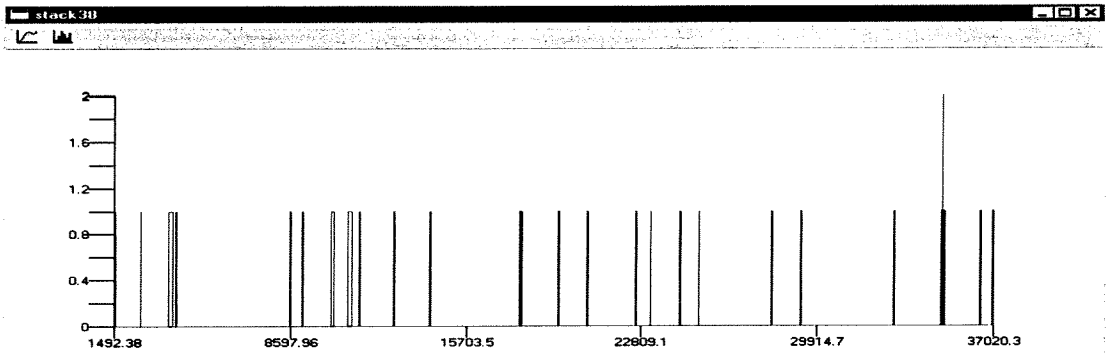


Figure 22.16, rgv row on warehouse lane 38

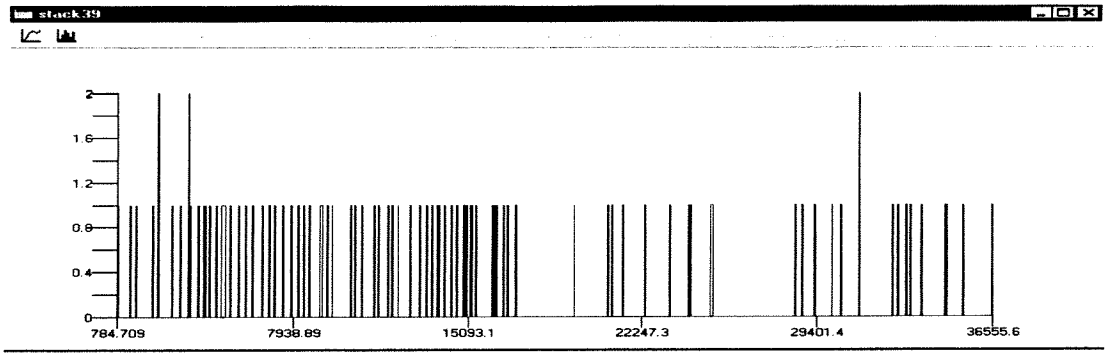


Figure 22.17, rgv row on warehouse lane 39

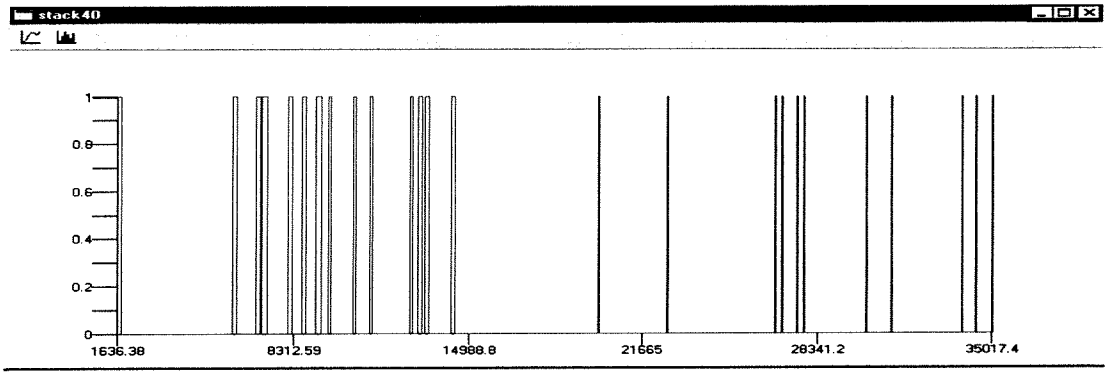


Figure 22.18, rgv row on warehouse lane 40

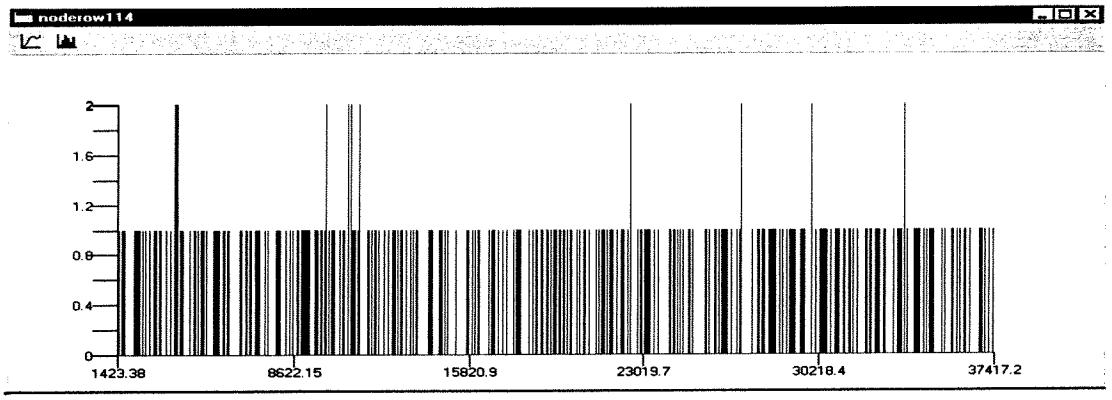


Figure 22.19, rgv row on node 114

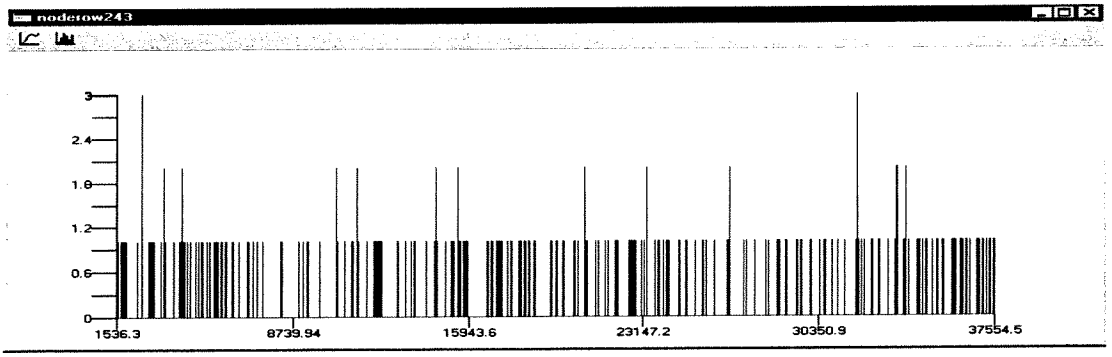


Figure 22.20, rgv row on node 243

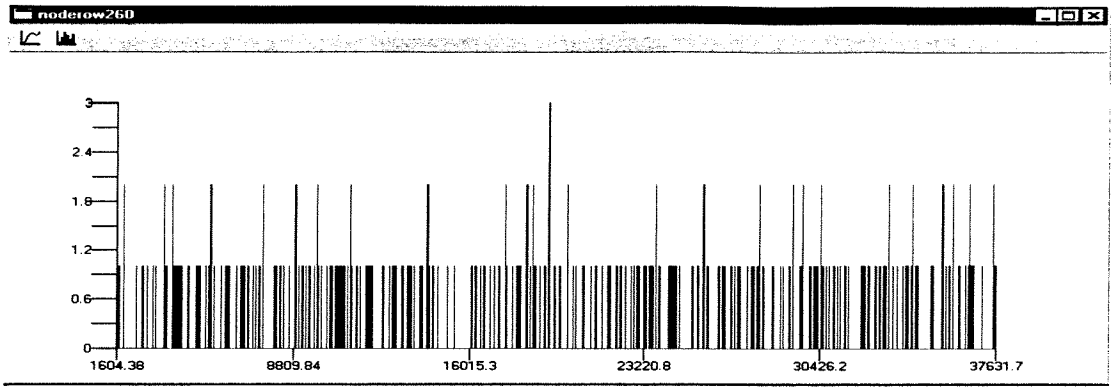


Figure 22.21, rgv row on node 260

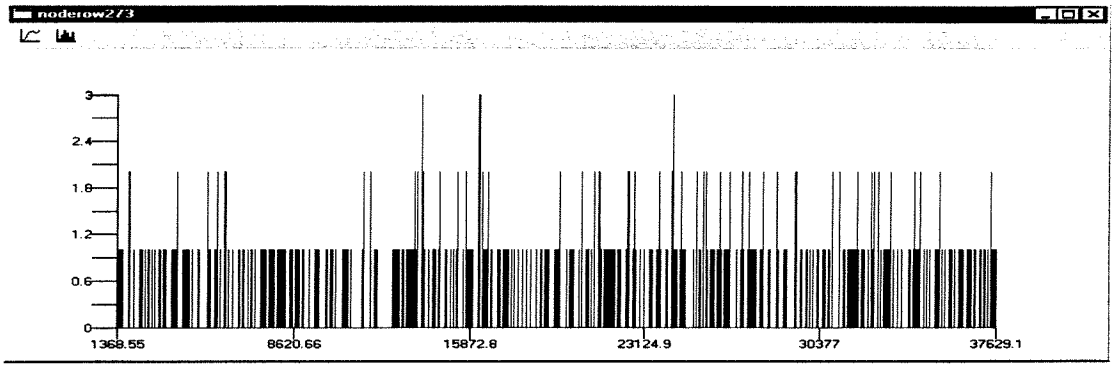


Figure 22.22, rgv row on node 273

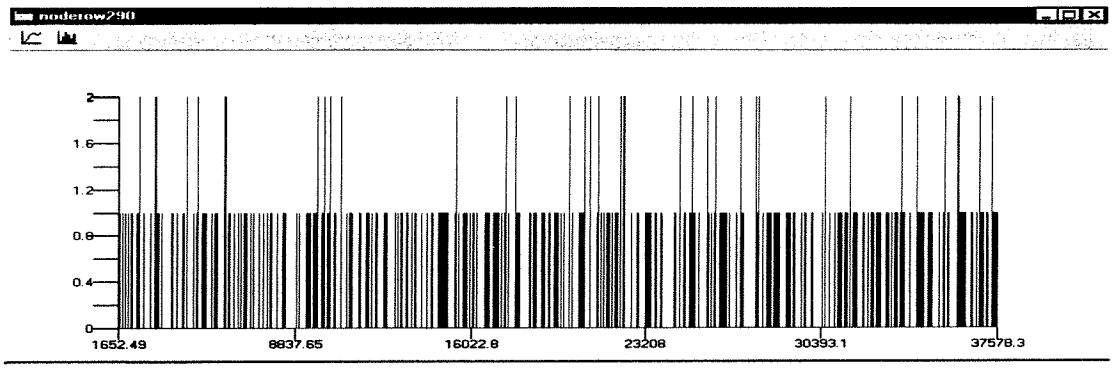


Figure 22.23, rgv row on node 290

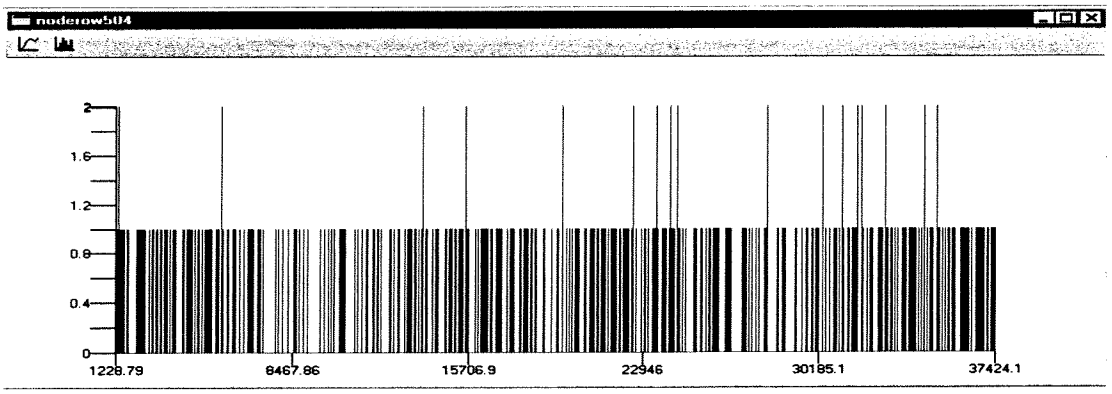


Figure 22.24, rgv row on node 504

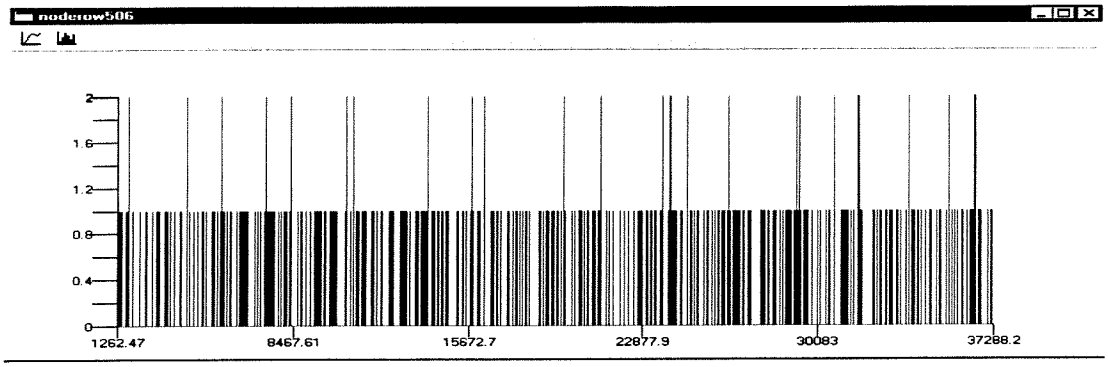


Figure 22.25, rgv row on node 506

Appendix 23: Input parameters for the experiment "shuffling from 3 berths"

The basic parameters used for this experiment are shown in Table 23.1. The parameters frombtob represent the containers that need to be shuffled between the yards. Inplacecouter 1, 2 and 3 contain some containers. These are to load the three vessels before the shuffling can start.
The parameter values after the run are shown in Table 23.2.

Parameter	Value	
frombtob[1,2]	1200	containers
frombtob[3,2]	1200	containers
frombtob[2,1]	1600	containers
frombtob[2,3]	1600	containers
inplacecounter 1	50	containers
inplacecounter 2	70	containers
inplacecounter 3	50	containers
seed of uniform	67	-
seed of arrivaltimes feeders	17	-
seed of arrivaltimes big	27	-
import generator feeders	40	containers
export generator feeders	40	containers
import generator big vessels	60	containers
export generator big vessels	60	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
rgv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	70	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	58	seconds

Table 23.1: Input parameters for experiment, "shuffling from 3 berths"

Parameter	Value	
frombtob[1,2]	0	containers
frombtob[3,2]	0	containers
frombtob[2,1]	0	containers
frombtob[2,3]	0	containers
Inplacecounter 1	1631	containers
Inplacecounter 2	2490	containers
Inplacecounter 3	1649	containers

Table 23.2, output after experiment, "Shuffling from 3 berths"

In Table 23.2 the parameters inplacecounter 1,2 and 3 indicate 5770 containers in total have been shuffled between the berths 1,2 and 3. Actually only 5740 containers have been shuffled, 10 containers behind each berth already where in place. Frombtob[1,2] =0 indicates there are no containers left in Yard 1 destined to go to Yard 2.

23.1.1 Warehouse cranes located in Yard 2 while shuffling on all three the berths

The cranes located in Yard 2 will have to deal with 5600 containers. From Yard 1 & 3 the rgv's will deliver 2 * 1200 = 2400 containers, while Yard 2 itself must deliver 3200 containers to Yard 1 & 3.

Considering that the number of containers that will be retrieved and delivered form the 20' stack and the 40' stacks are the same means that 2800 containers will be handled by both stacks types.

- Required 20' warehouse crane operational time: 2800 * 80 seconds = 224,000 seconds
- Required 40' warehouse crane operational time: 2800 * 80 seconds = 224,000 seconds

Number of warehouse cranes in Yard 2:
- Number of warehouse cranes in the 20' stacks in Yard 2 = 6 * 2 = 12 warehouse cranes
- Number of warehouse cranes in the 40' stacks in Yard 2 = 6 * 3 = 18 warehouse cranes

Needed operational time per warehouse crane:

- $224{,}000/12 = 18{,}667$ seconds per 20' stack warehouse crane
- $224{,}000/18 = 12{,}444$ seconds per 40' stack warehouse crane

Required adjacent loading time for both feeder vessels: 6.5 hours $\Rightarrow$ 23400 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(18{,}667/23{,}400\right) * 100\% = 79.8\ \%$

40' stack: $\left(12{,}444/23{,}400\right) * 100\% = 53.2\ \%$

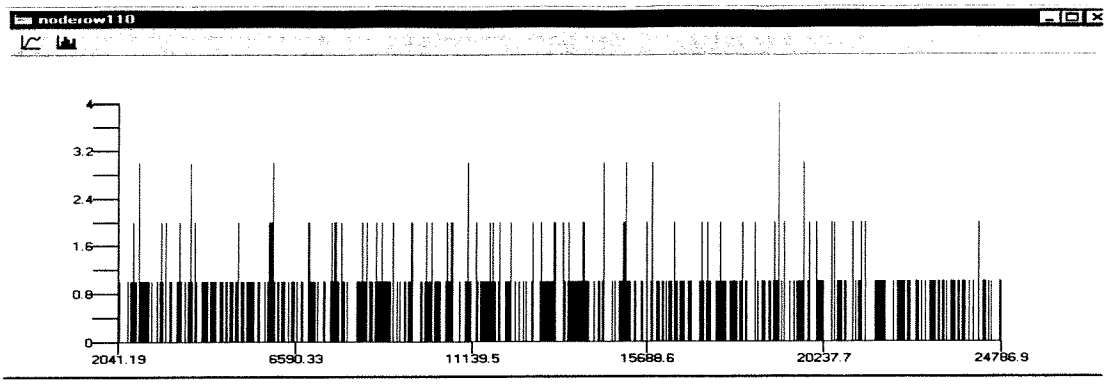


Figure 23.1, rgv queue at node 110

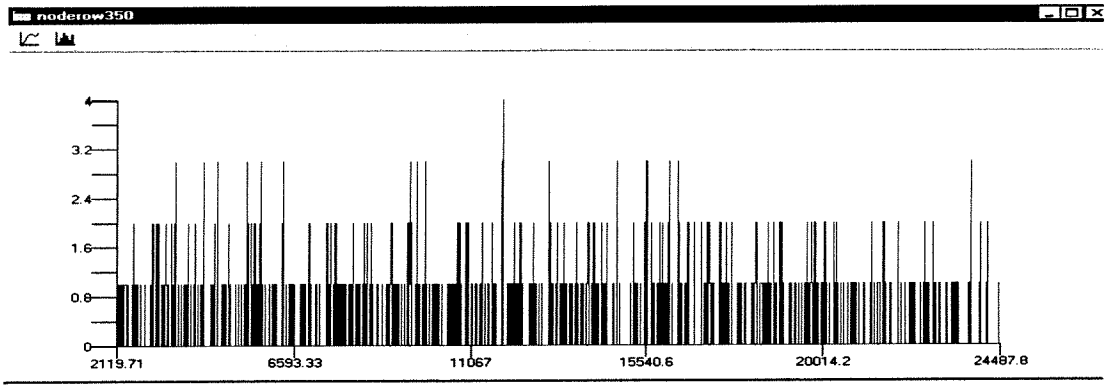


Figure 23.2, rgv queue at node 350

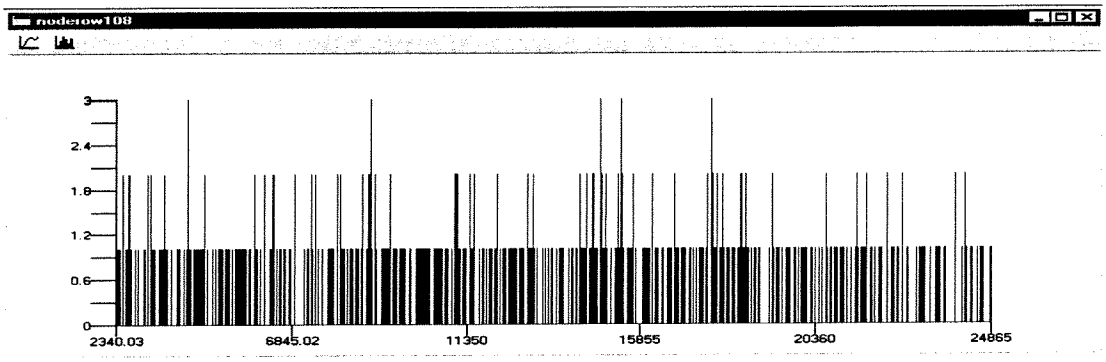


Figure 23.3, rgv queue at node 108

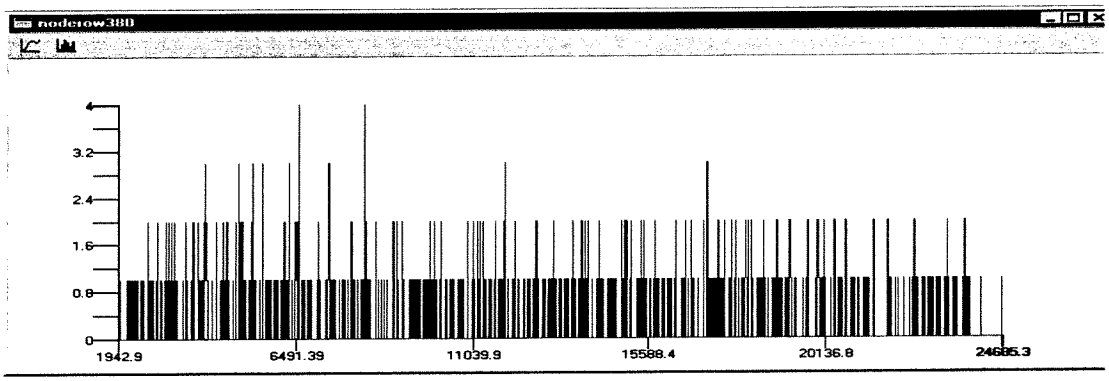


Figure 23.4, rgv queue at node 380

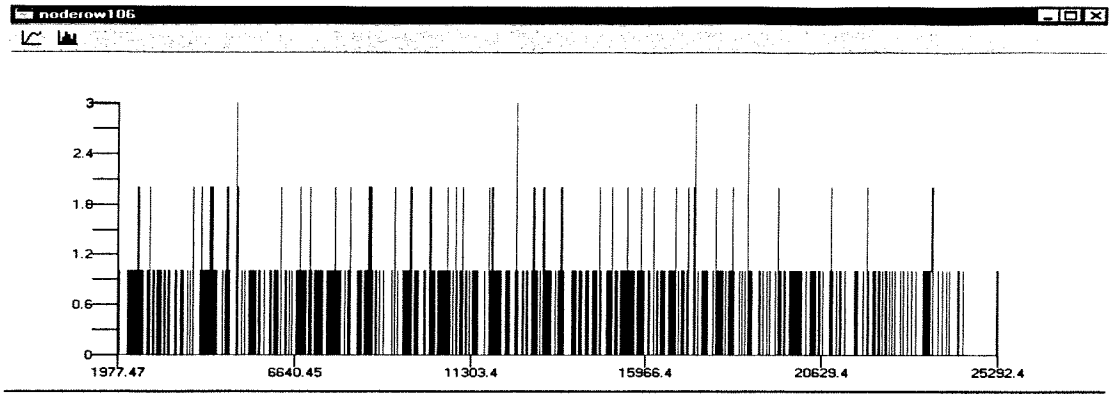


Figure 23.5, rgv queue at node 106

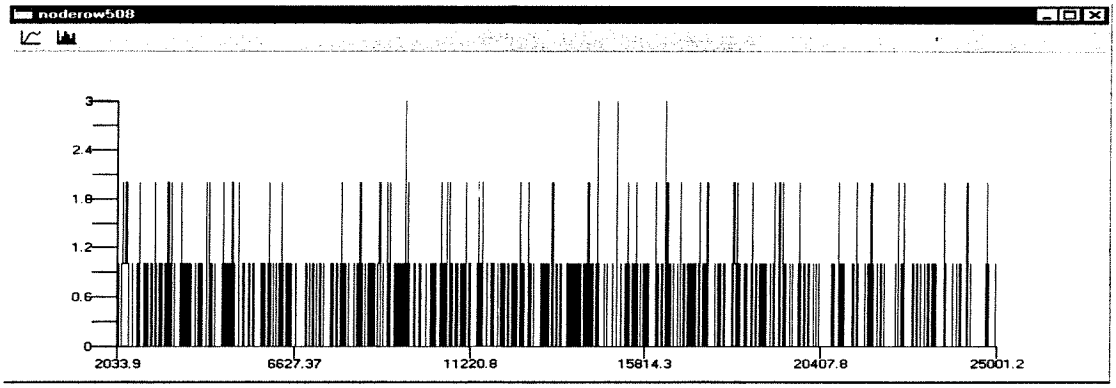


Figure 23.6, rgv queue at node 508

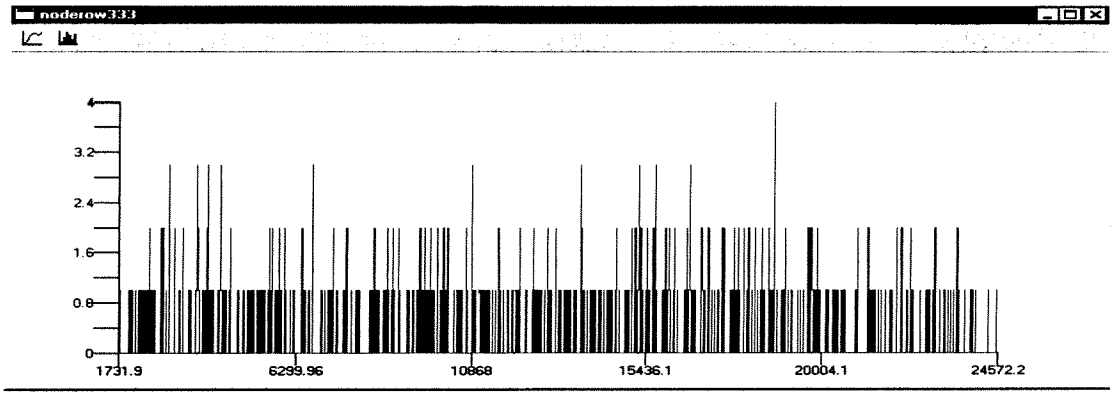


Figure 23.7, rgv queue at node 333



Figure 23.8, rgv queue at node 510

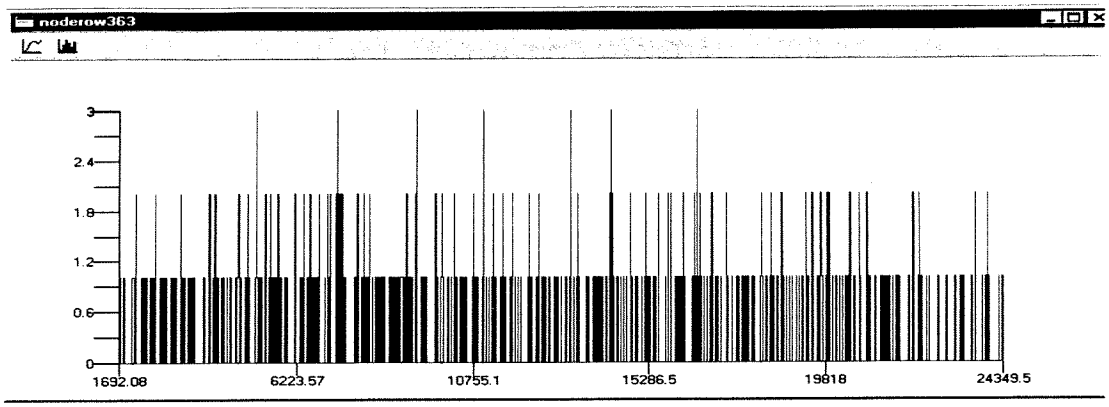


Figure 23.9, rgv queue at node 363

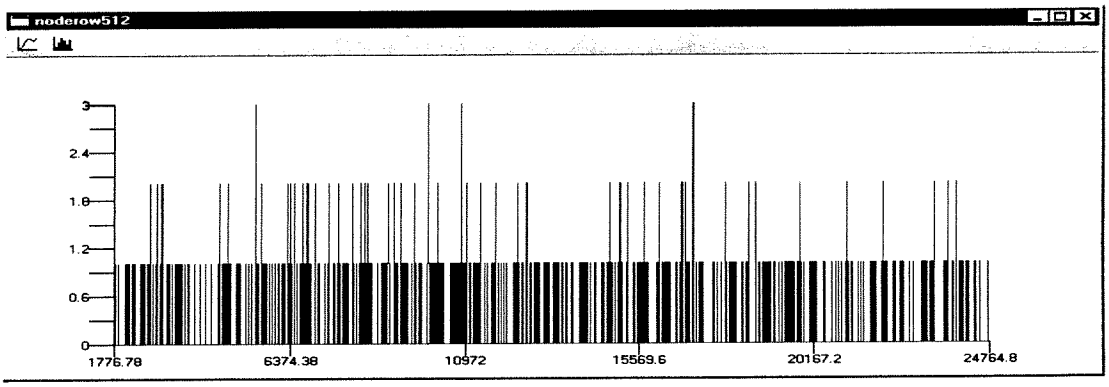


Figure 23.10, rgv queue at node 512

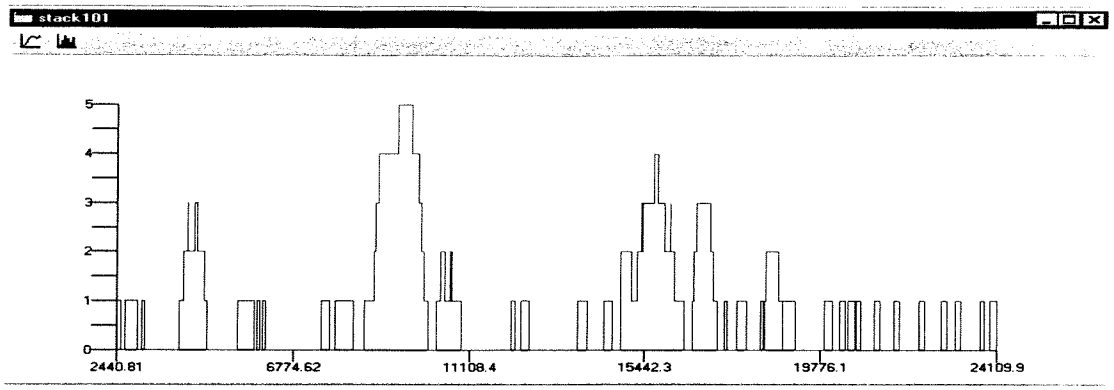


Figure 23.11, rgv queue in warehouse lane 101

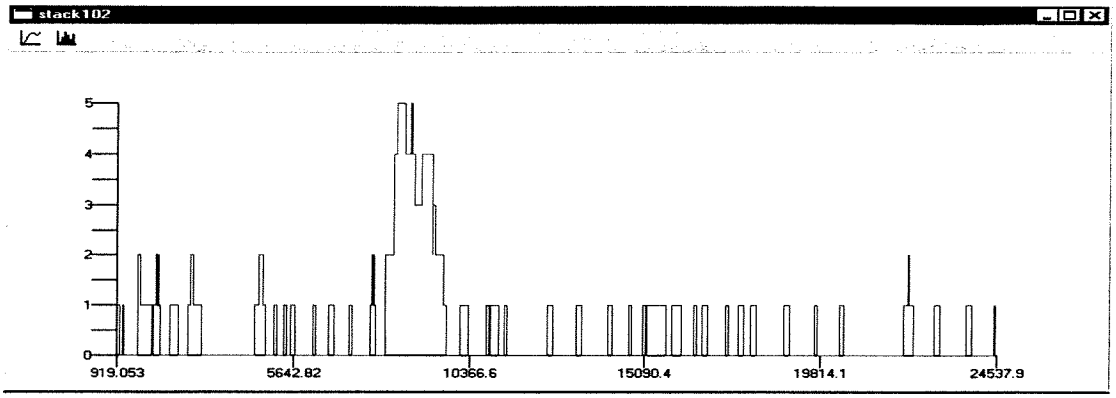


Figure 23.12, queue in warehouse lane 102

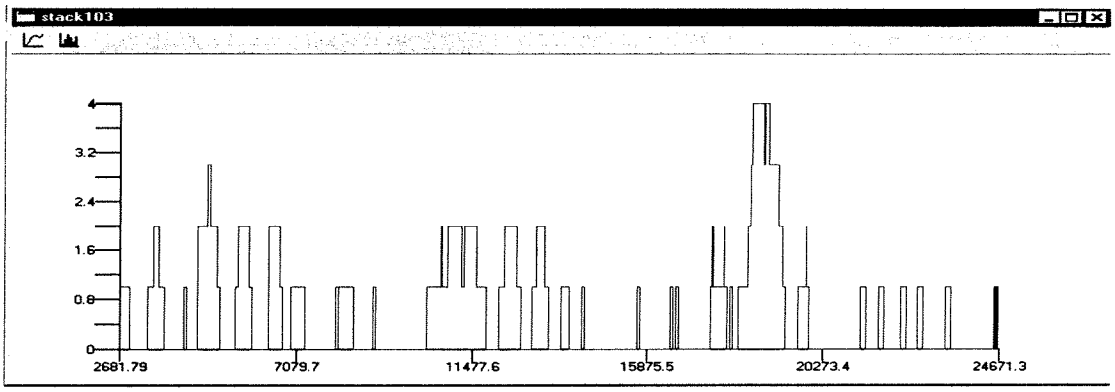


Figure 23.13, queue in warehouse lane 103

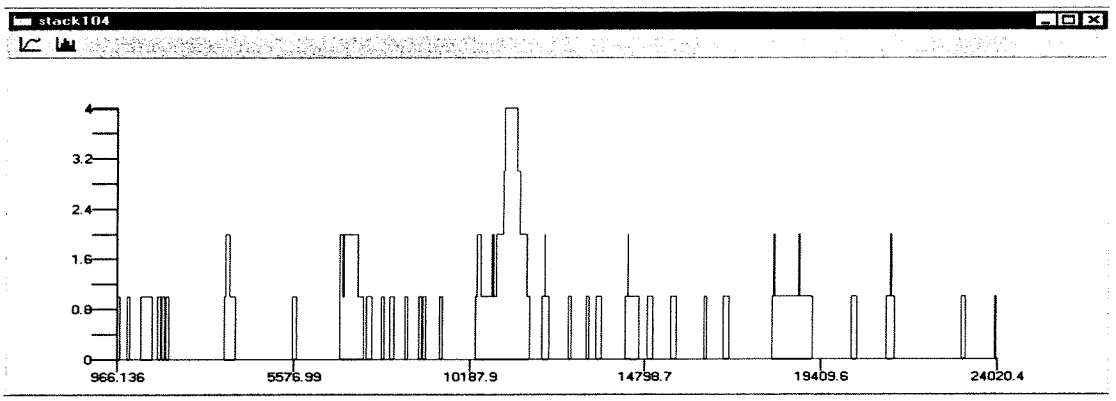


Figure 23.14, queue in warehouse lane 104

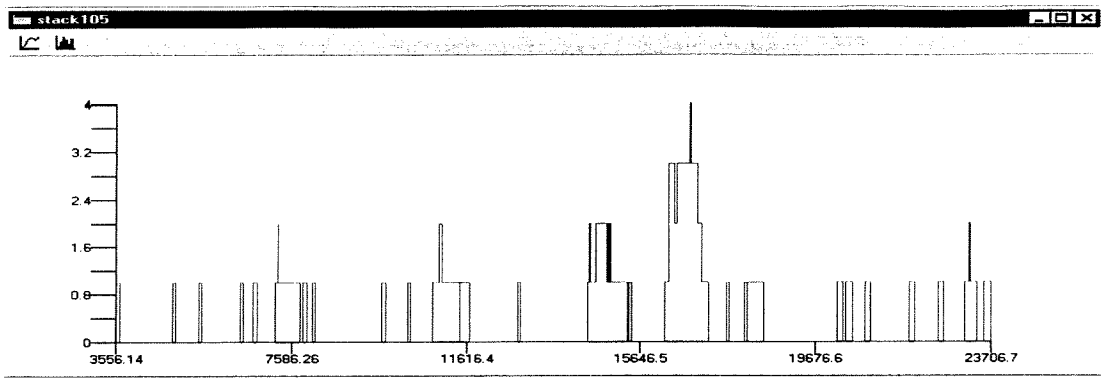


Figure 23.15, queue in warehouse lane 105

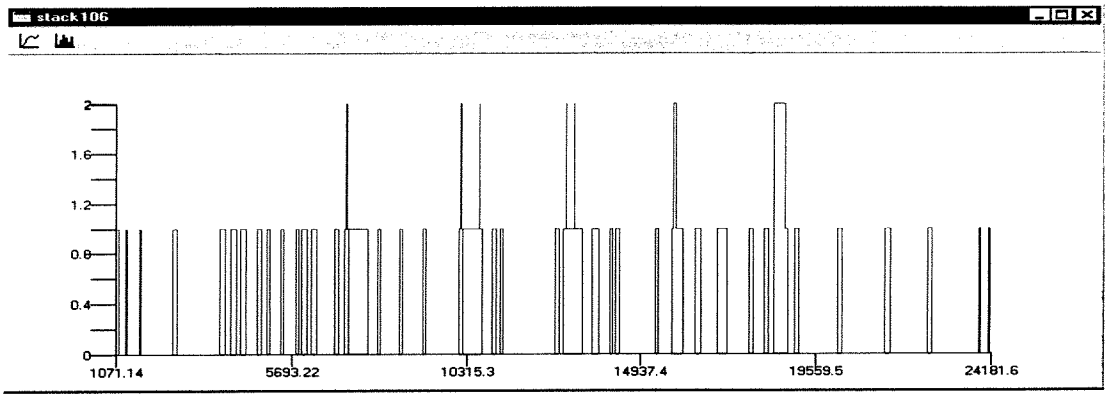


Figure 23.16, queue in warehouse lane 106

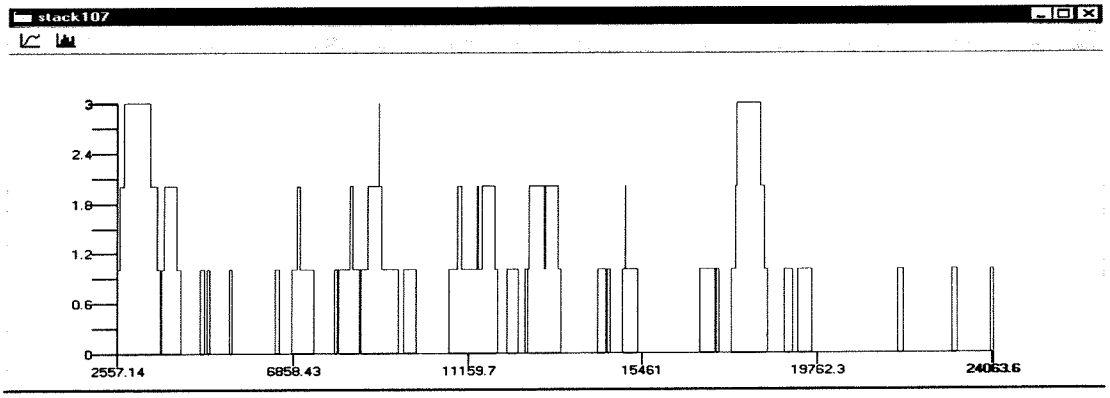


Figure 23.17, queue in warehouse lane 107

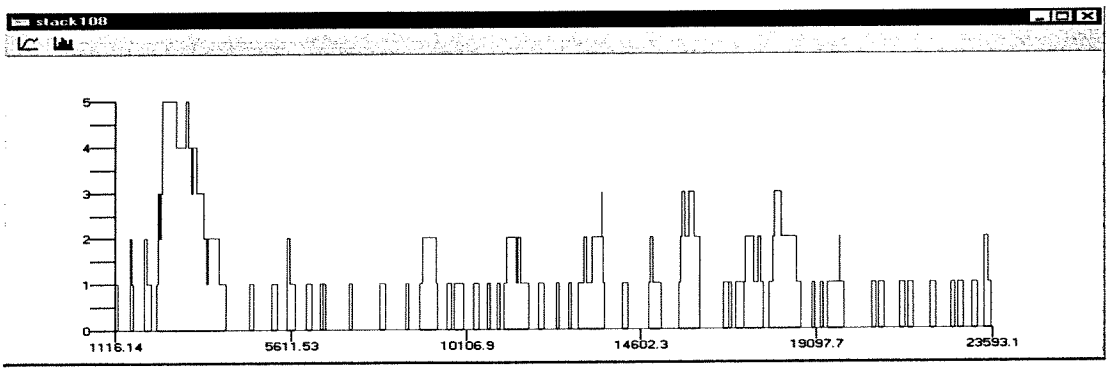


Figure 23.18, queue in warehouse lane 108

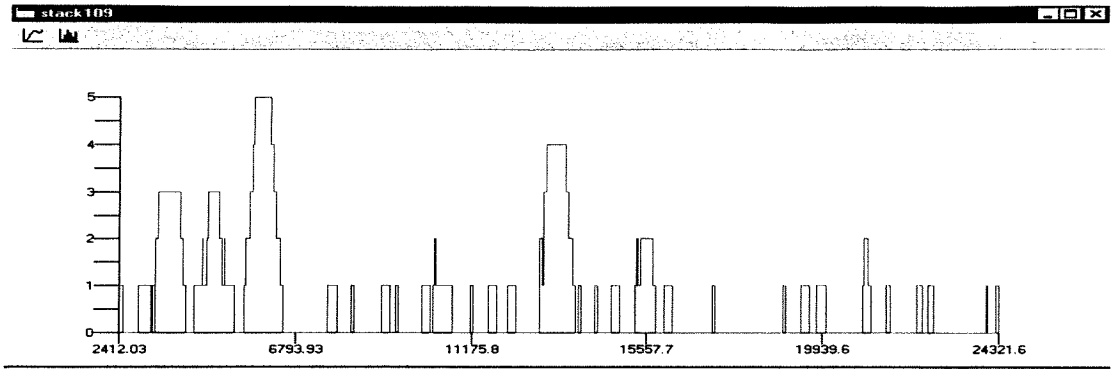


Figure 23.19, queue in warehouse lane 109

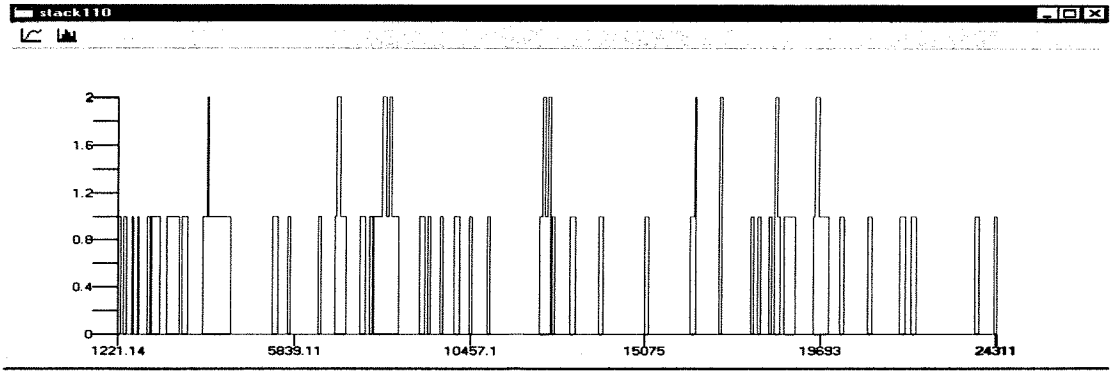


Figure 23.20, queue in warehouse lane 110

Appendix 24: Input parameters for the experiment adjacent loading at Berth 1 and 3 using 12 rgv's and unloading at Berth 2.

The basic parameters used for this experiment are shown in Table 24.1. The parameters are the same as used in experiment 10, except for the number of rgv's used when loading a vessel from an adjacent berth. In this experiment that is the case for Berth 1 and Berth 3.

Parameter	Value	
frombtob[1,2]	0	containers
frombtob[3,2]	0	containers
frombtob[2,1]	1600	containers
frombtob[2,3]	1600	containers
inplacecounter 1	0	containers
inplacecounter 2	0	containers
inplacecounter 3	0	containers
seed of uniform	67	-
seed of arrivaltimes feeders	17	-
seed of arrivaltimes big	27	-
import generator feeders	40	containers
export generator feeders	1050	containers
import generator big vessels	3150	containers
export generator big vessels	0	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
rgv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	70	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	**12**	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	58	seconds

Table 24.1 input parameters for experiment; Adjacent loading at Berth 1 and Berth 3 using 12 rgv's and unloading at Berth 2.

Parameter	Value	
frombtob[1,2]	40	containers
frombtob[3,2]	40	containers
frombtob[2,1]	2083	containers
frombtob[2,3]	2087	containers

Table 24.2 output parameter values for experiment; Adjacent loading at Berth 1 and Berth 3 using 12 rgv's and unloading at Berth 2.

24.1.1 Adjacent loading on Berths 1& 3 with 12 rgv's per portainer; unloading on Berth 2

As both feeder vessels will be provided from Yard 2 this means that 1050 * 2 = 2100 containers have to retrieved from these stacks. Both feeder vessels will leave the terminal after 4.3 hours. In this time span the mega vessel has unloaded 1730 containers into Yard 2. Adding these up to the 2100 containers loaded onto both feeder vessels will amount to 3830 containers.

Considering that the number of containers that will retrieved from the 20' stacks and the 40' stacks are the same means that 1915 containers will be retrieved from both stack types.

- Required 20' warehouse crane operational time: 1915 * 80 seconds = 153,200 seconds
- Required 40' warehouse crane operational time: 1915 * 80 seconds = 153,200 seconds

Number of warehouse cranes in Yard 2:
- Number of warehouse cranes in the 20' stacks in Yard 2 = 6 * 2 = 12 warehouse cranes
- Number of warehouse cranes in the 40' stacks in Yard 2 = 6 * 3 = 18 warehouse cranes

Needed operational time per warehouse crane:

- $153{,}200/12 = 12{,}767$ seconds per 20' stack warehouse crane
- $153{,}200/18 = 8{,}511$ seconds per 40' stack warehouse crane

Required adjacent loading time for both feeder vessels: 4.7 hours $\Rightarrow$ 16,920 seconds

Occupation rates of the warehouse cranes:

20' stack: $\left(12{,}767/16{,}920\right) * 100\% = 75.5\,\%$

40' stack: $\left(8{,}511/16{,}920\right) * 100\% = 50.3\,\%$

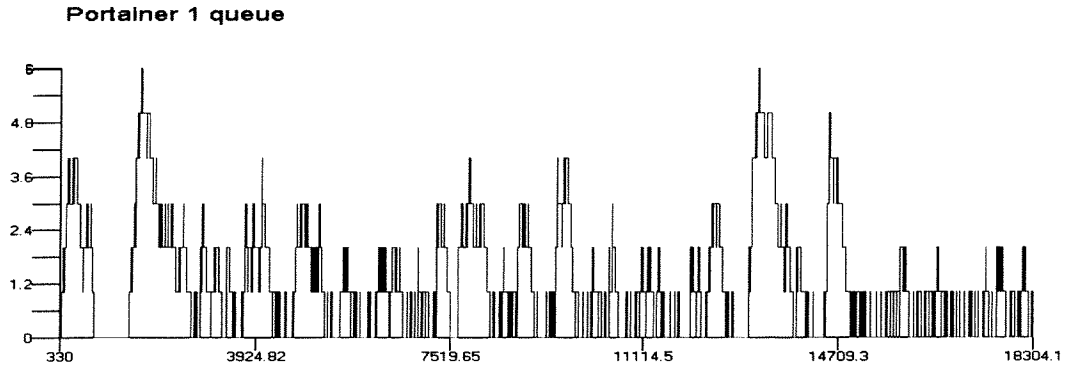


Figure 24.1, rgv row at portainer 1

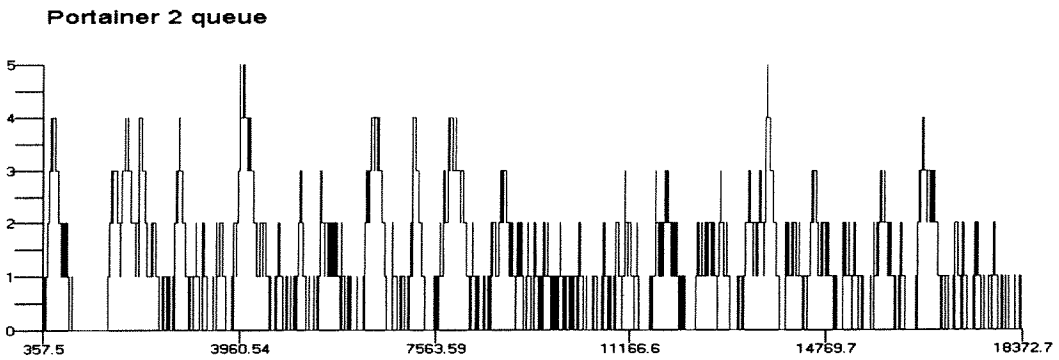


Figure 24.2, rgv row at portainer 2

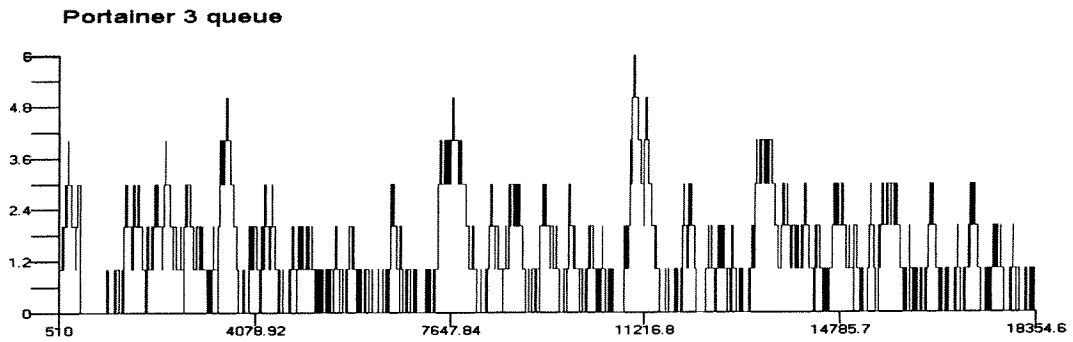


Figure 24.3, rgv row at portainer 3

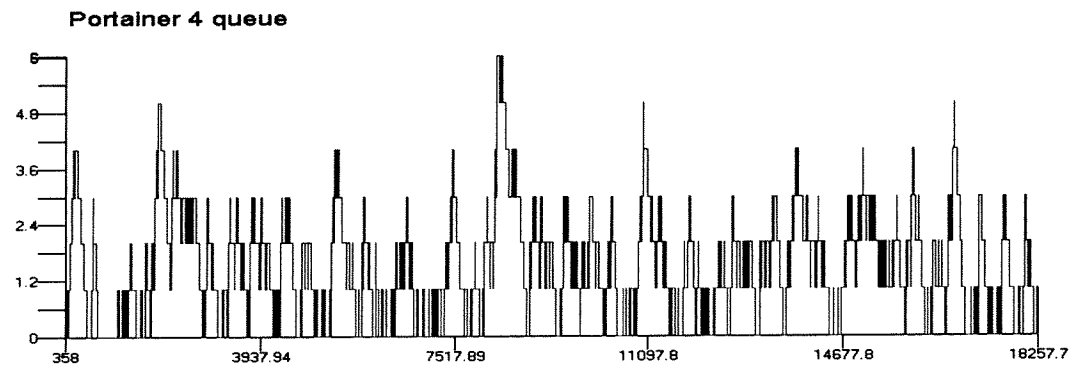


Figure 24.4, rgv row at portainer 4

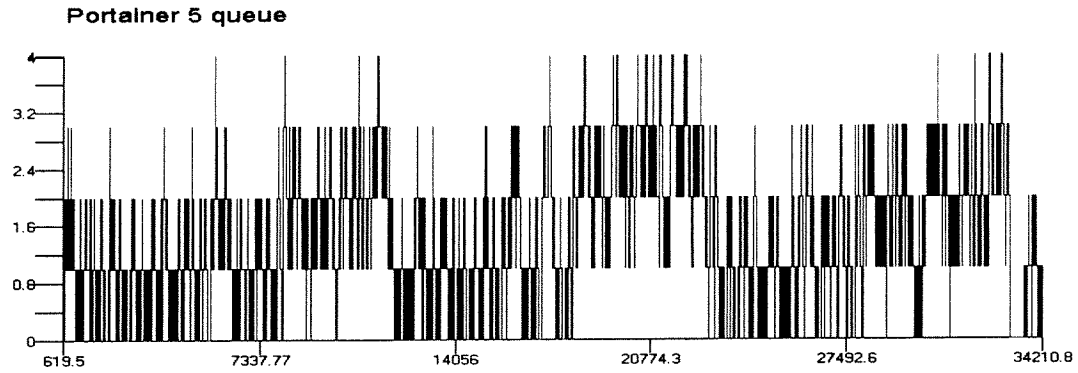


Figure 24.5, rgv row at portainer 5

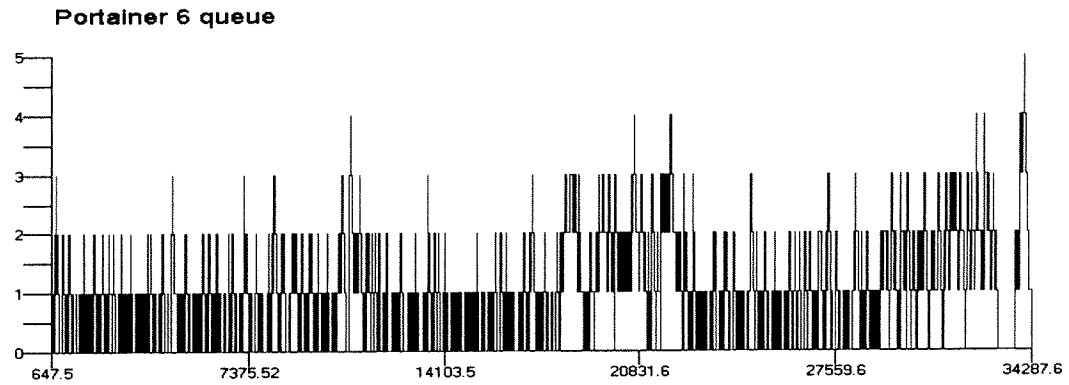


Figure 24.6, rgv row at portainer 6

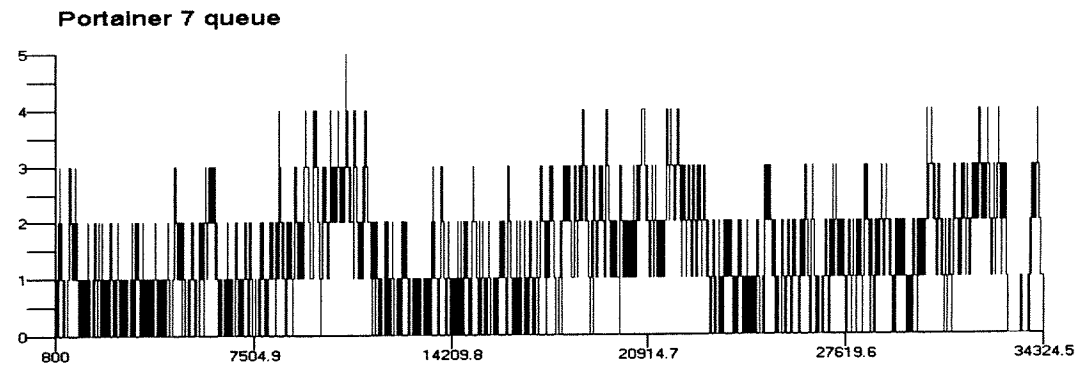


Figure 24.7, rgv row at portainer 7

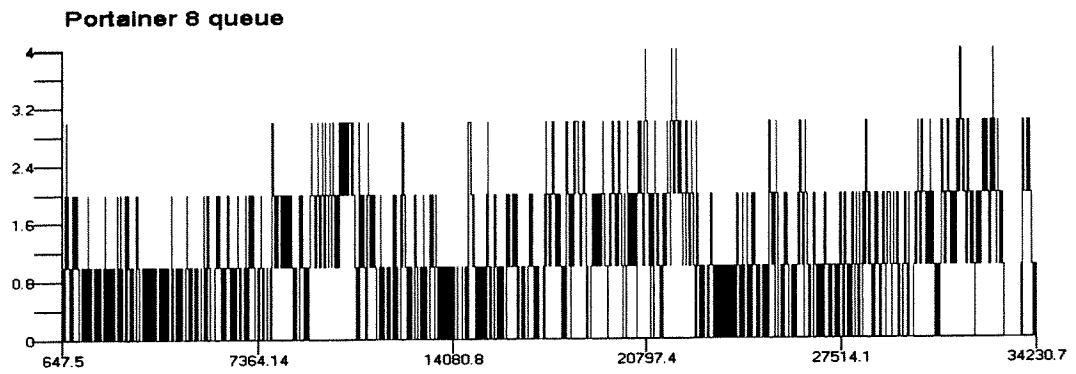


Figure 24.8, rgv row at portainer 8

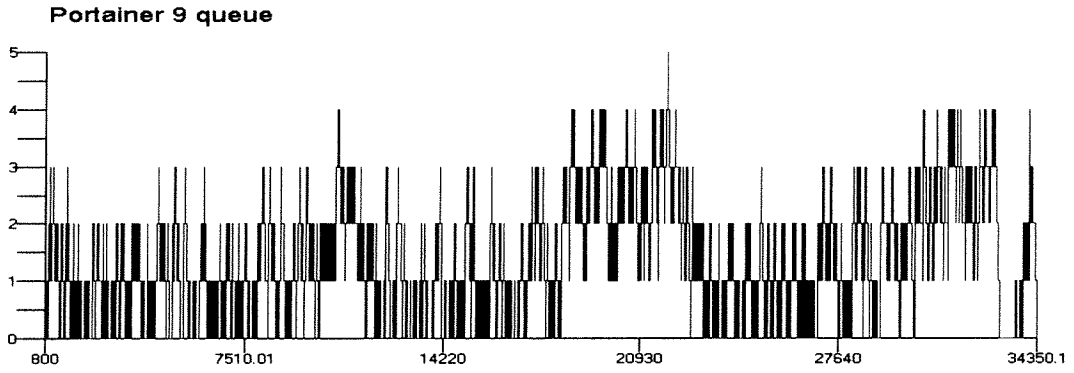


Figure 24.9 , rgv row at portainer 9

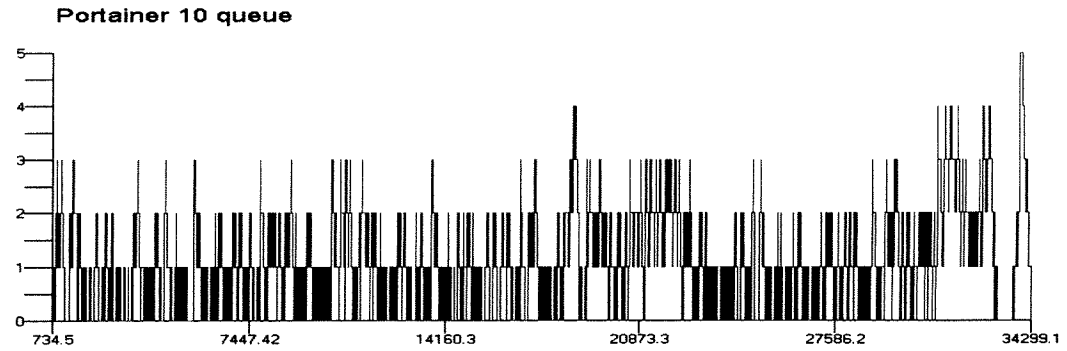


Figure 24.10, rgv row at portainer 10

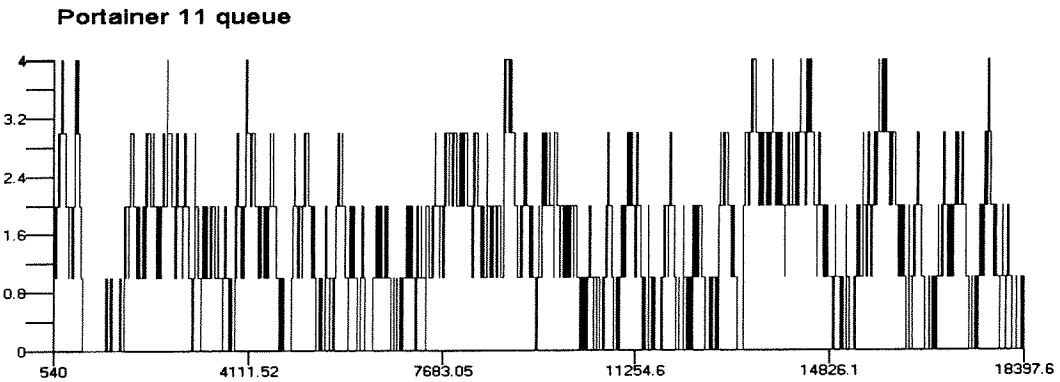


Figure 24.11, rgv row at portainer 11

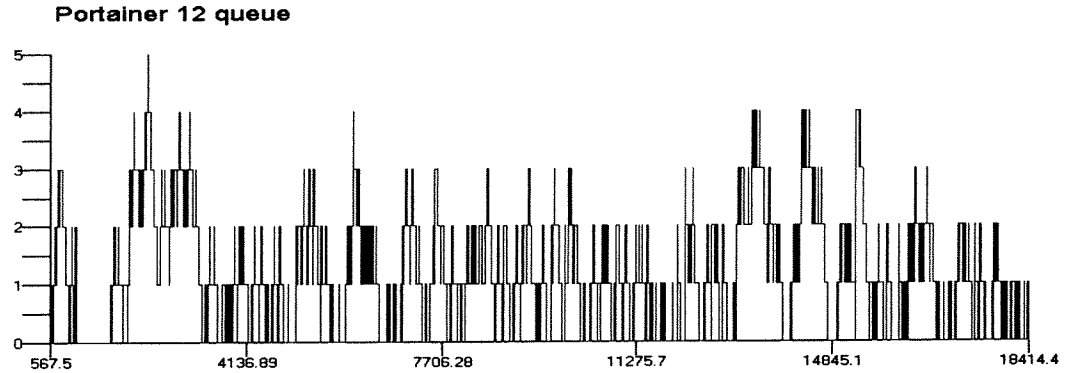


Figure 24.12, rgv row at portainer 12

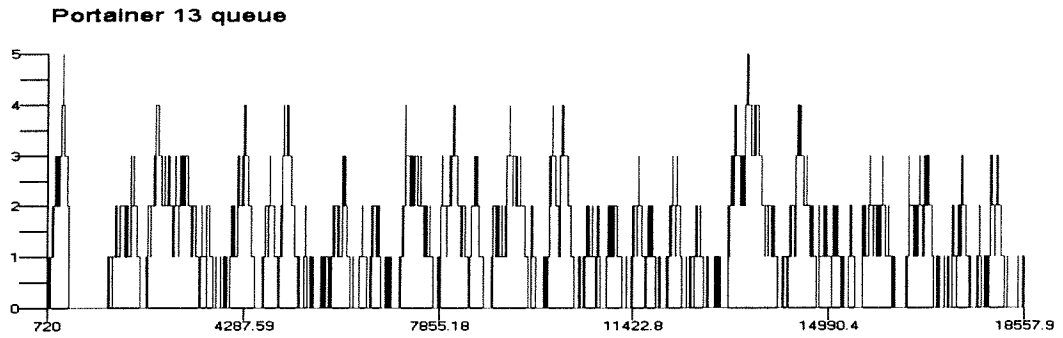


Figure 24.13, rgv row at portainer 13

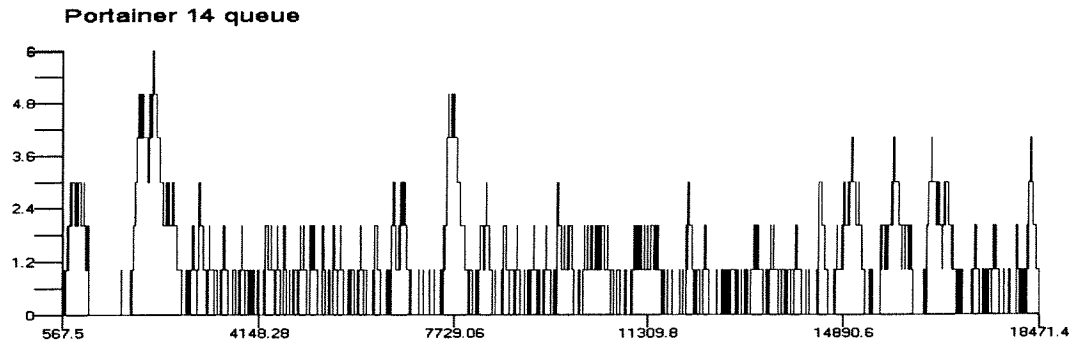


Figure 24.14, rgv row at portainer 14

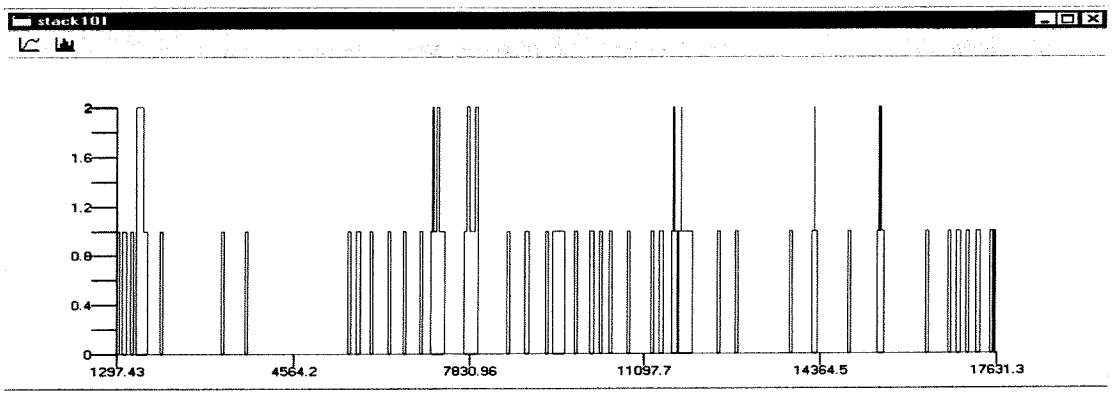


Figure 24.15, rgv stack row 101

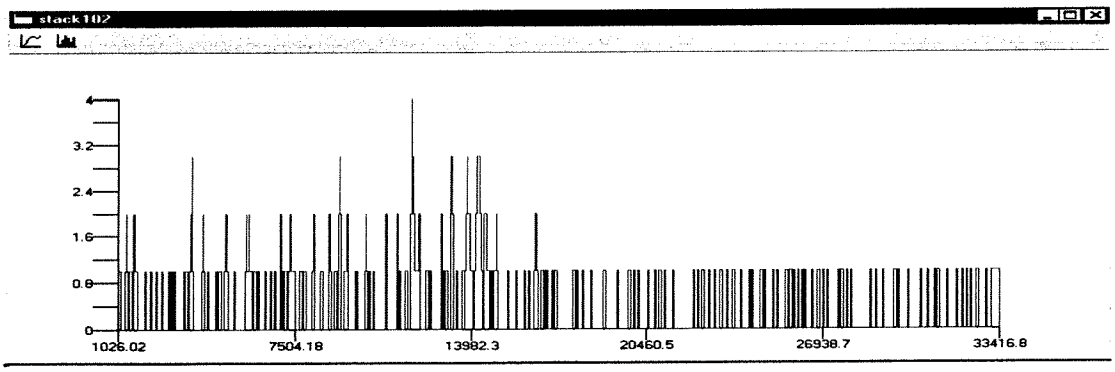


Figure 24.16, rgv stack row 102

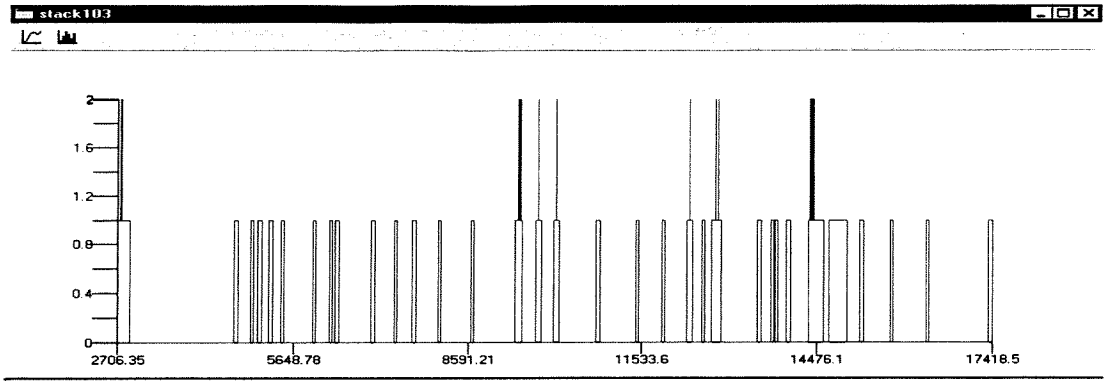


Figure 24.17, rgv stack row 103

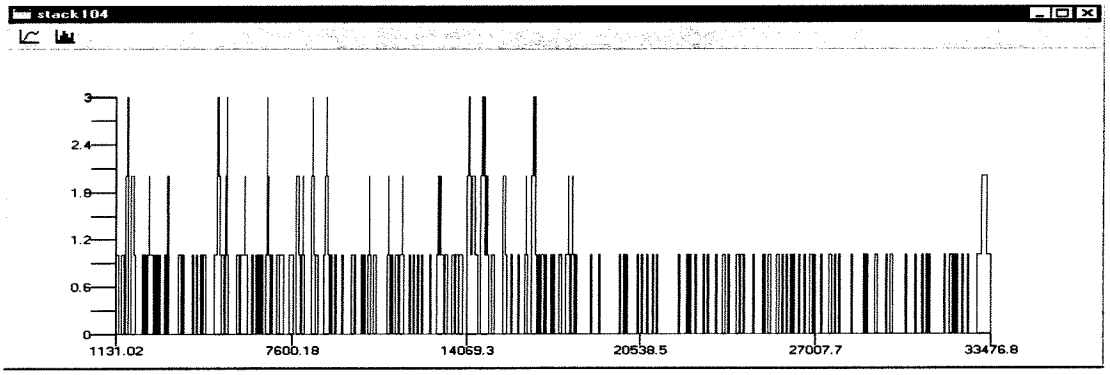


Figure 24.18, rgv stack row 104

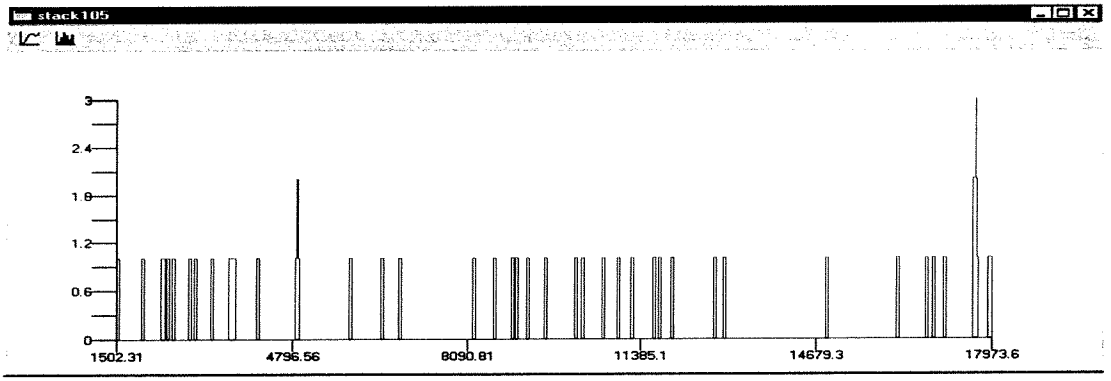


Figure 24.19, rgv stack row 105

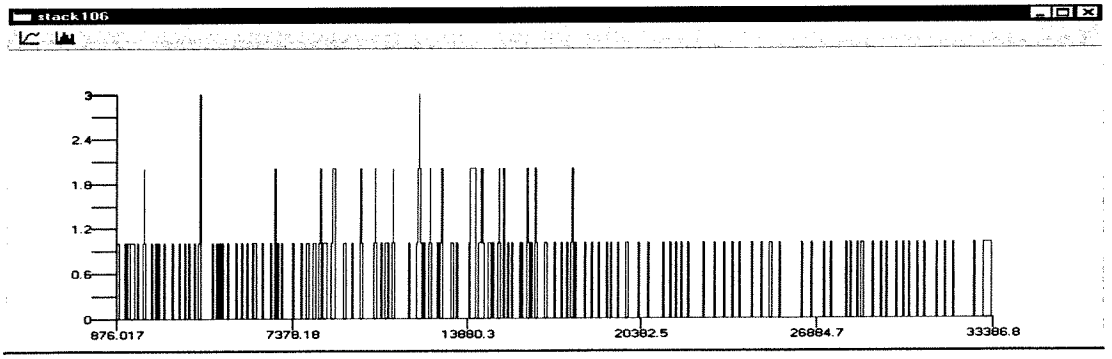


Figure 24.20, rgv stack row 106

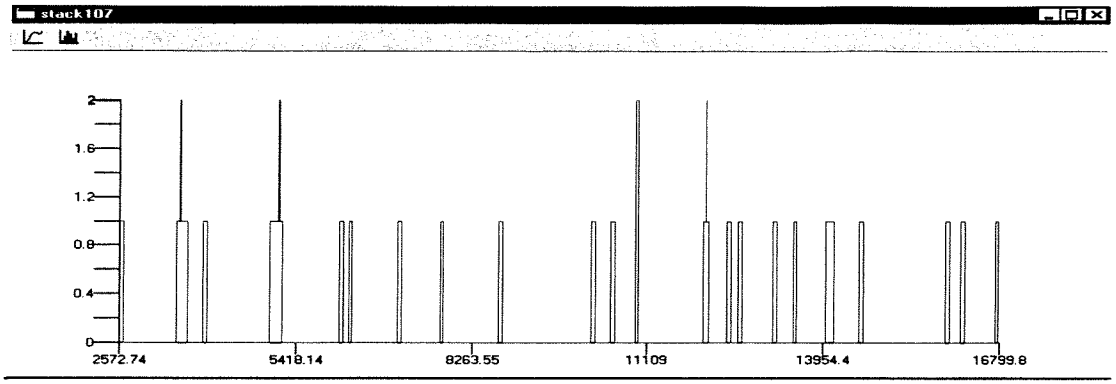


Figure 24.21, rgv stack row 107

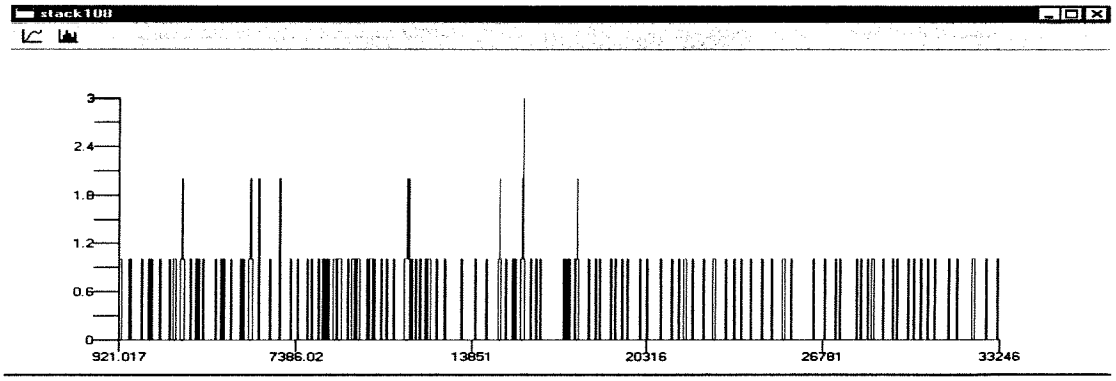


Figure 24.22, rgv stack row 108

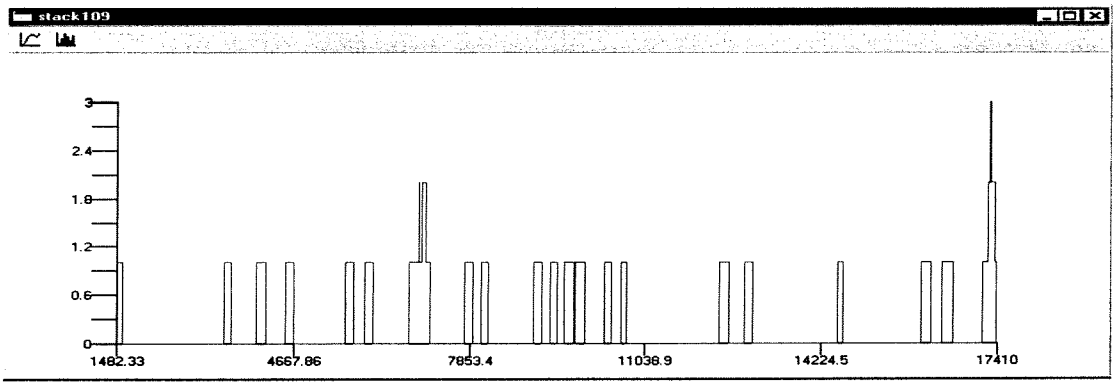


Figure 24.23, rgv stack row 109

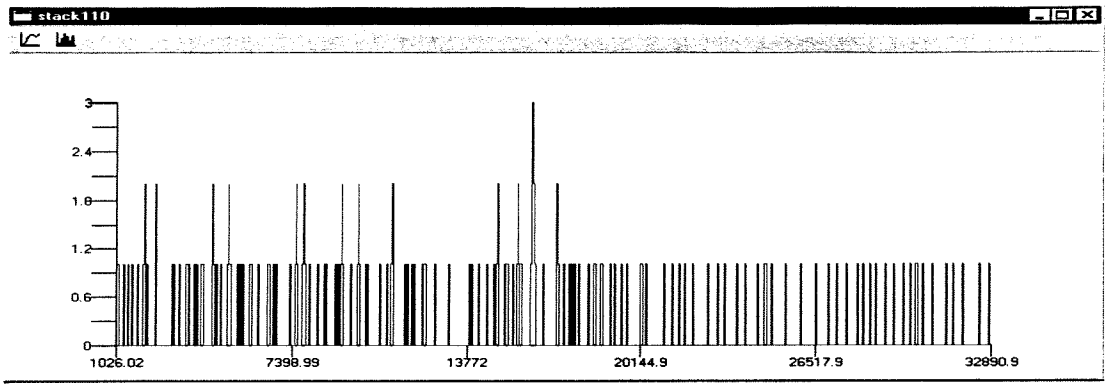


Figure 24.24, rgv stack row 110



Figure 24.25, node row 303

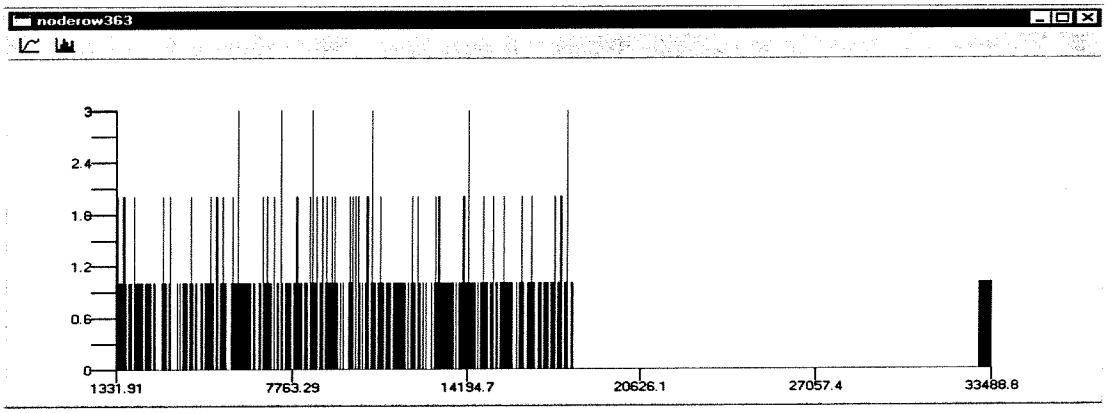


Figure 24.26, rgv node row 363

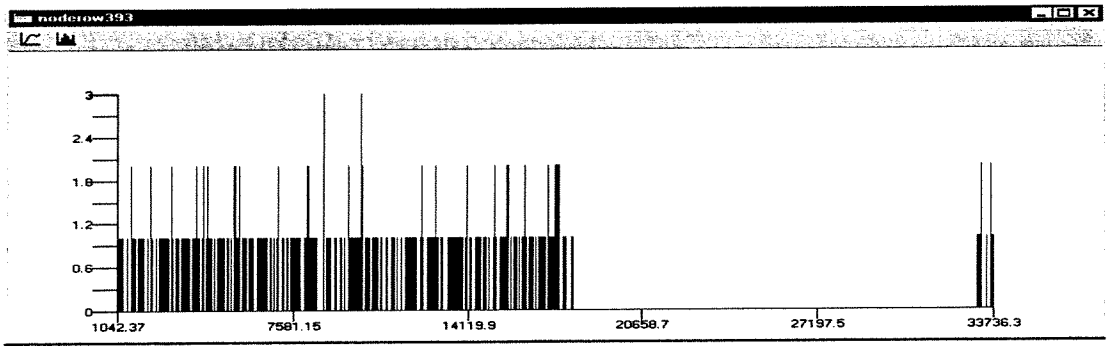


Figure 24.27, rgv node row 393

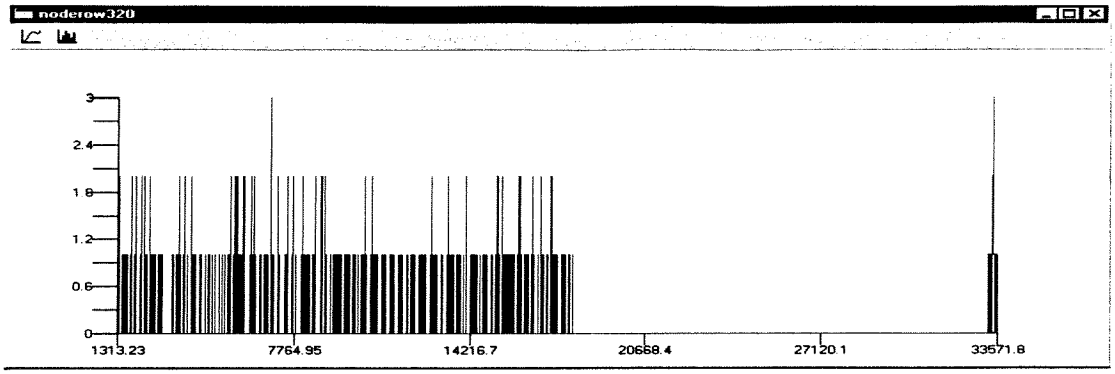


Figure 24.28, rgv node row 320

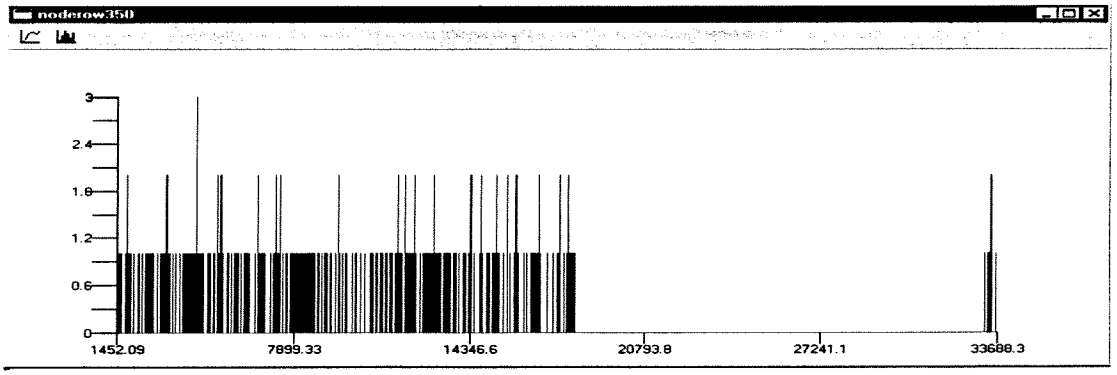


Figure 24.29, rgv node row 350

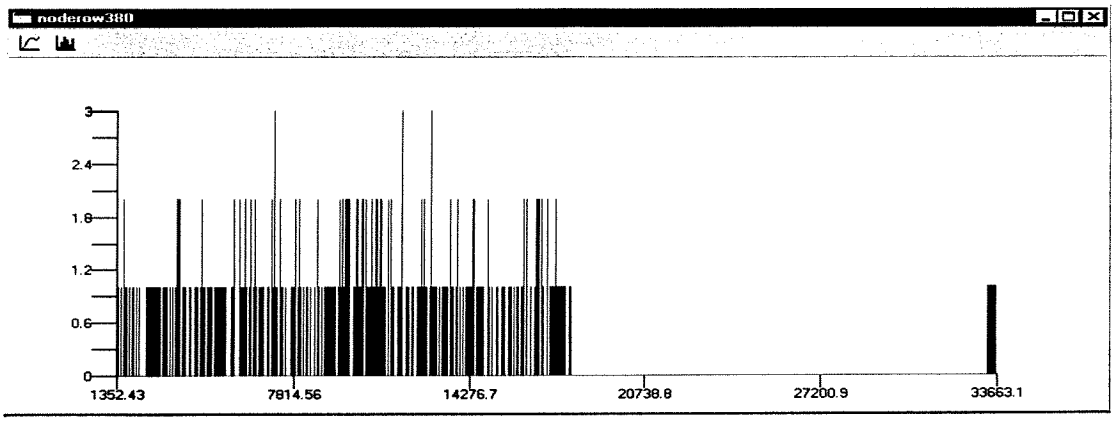


Figure 24.30, rgv node row 504

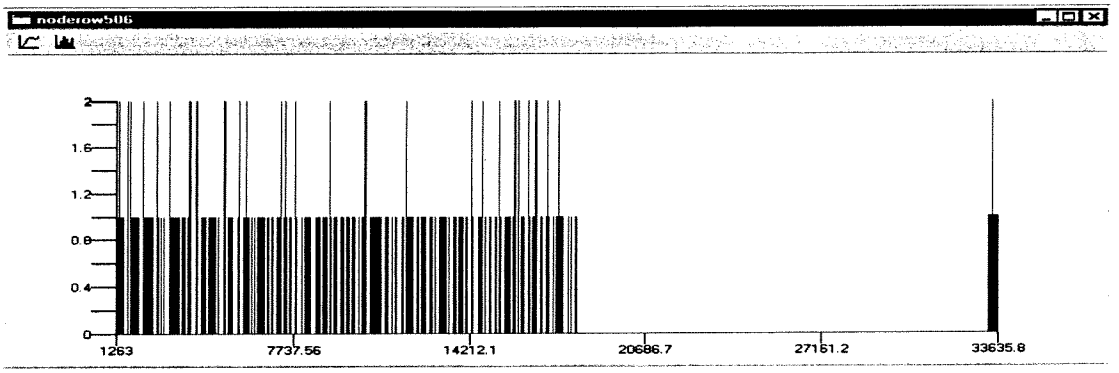


Figure 24.31, rgv node row 506

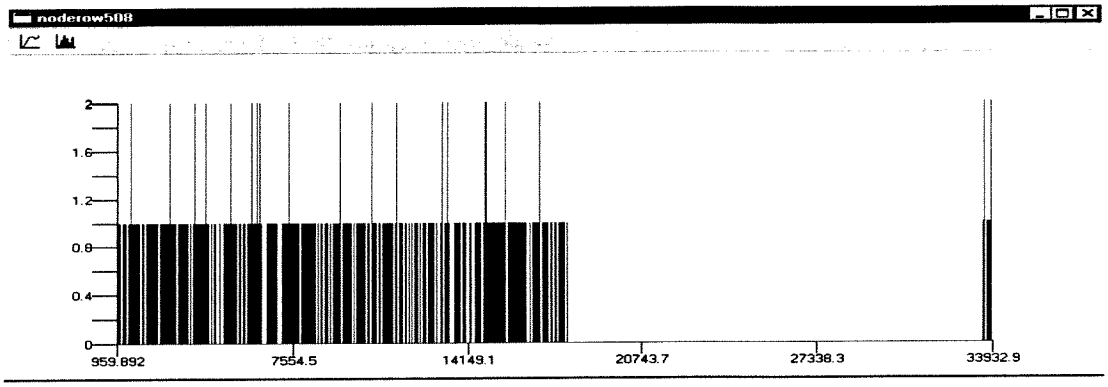


Figure 24.32, rgv node row 508

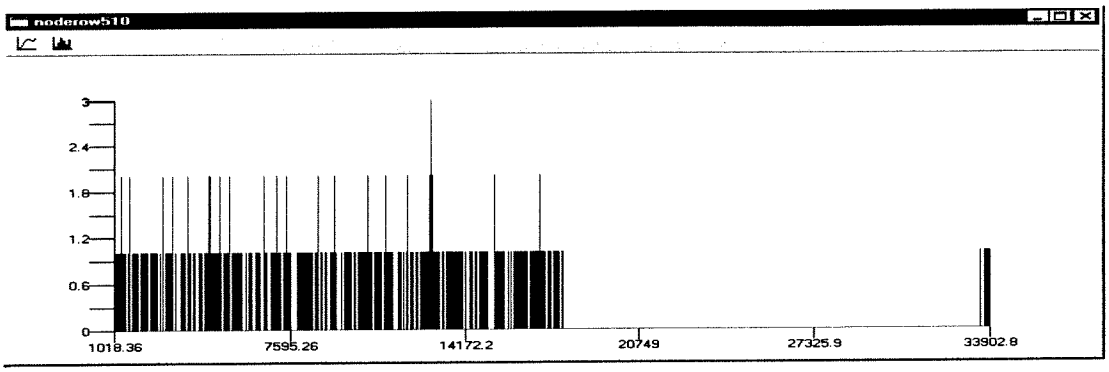


Figure 24.33, rgv node row 510



Figure 24.34, rgv node row 512

Appendix 25: Input parameters for the experiment with disturbances in the portainer cycle times.

The basic parameters used for this experiment are shown in Table 25.1. The parameters are the same as used in experiment 10, except for the portainer cycle times. The portainer cycle times used at the reference terminal are the shortest cycle times possible. In reality these cycletimes will not be as perfect. Because that would mean the portainer driver does not make any mistakes which would decrease the cycle times of the portainer. Therefore extra time is added to model disturbances. Two kinds of disturbances are added. A small disturbance with a maximum of 5 seconds and a big disturbance that only occurs once in a hundred moves.
Including these two disturbances, the new mean portainer cycle time is 63 seconds.

Parameter	Value	
frombtob[1,2]	0	containers
frombtob[3,2]	0	containers
frombtob[2,1]	1600	containers
frombtob[2,3]	1600	containers
inplacecounter 1	0	containers
inplacecounter 2	0	containers
inplacecounter 3	0	containers
seed of uniform	67	-
seed of arrivaltimes feeders	17	-
seed of arrivaltimes big	27	-
import generator feeders	40	containers
export generator feeders	1050	containers
import generator big vessels	3150	containers
export generator big vessels	0	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
rgv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	90	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
stacking crane cycle time	80	seconds
mean portainer cycle time	**63**	seconds

Table 25.1 input parameters for experiment; Adjacent loading at Berth 1 and Berth 3 and unloading at Berth 2, using variable portainer cycle times.

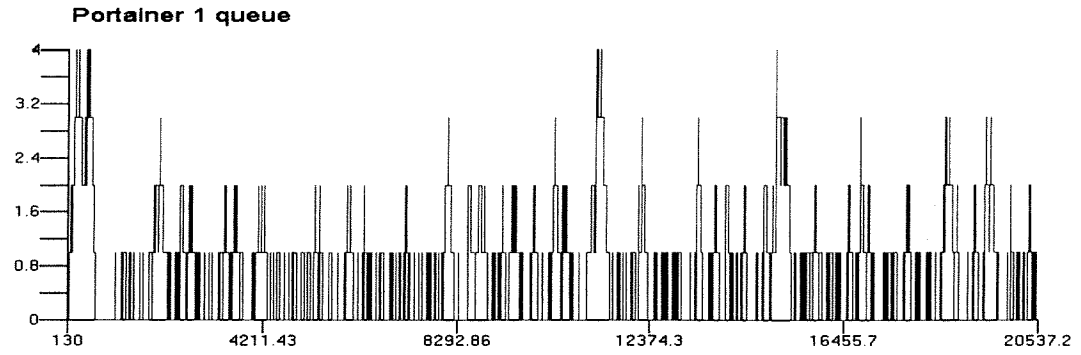


Figure 25.1, queue under portainer 1 while loading a feeder from an adjacent berth

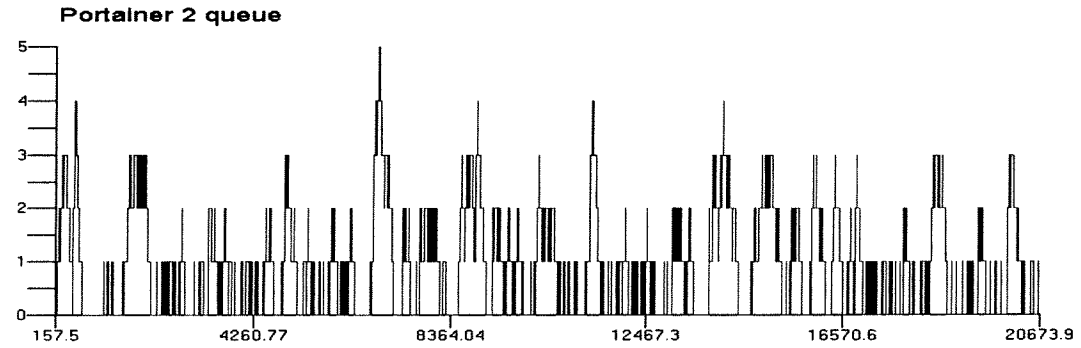


Figure 25.2, queue under portainer 2 while loading a feeder from an adjacent berth

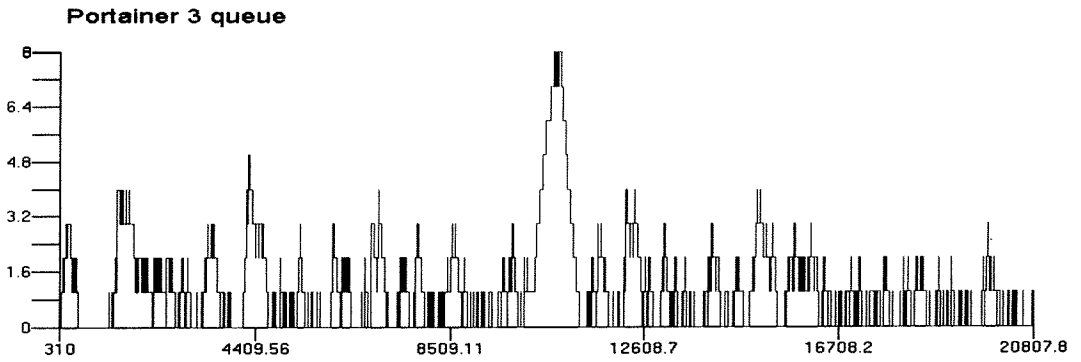


Figure 25.3, queue under portainer 3 while loading a feeder from an adjacent berth

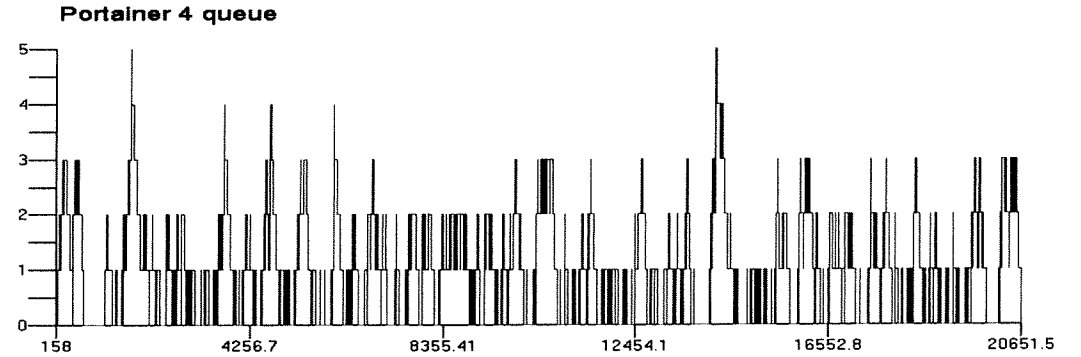


Figure 25.4, queue under portainer 4 while loading a feeder from an adjacent berth

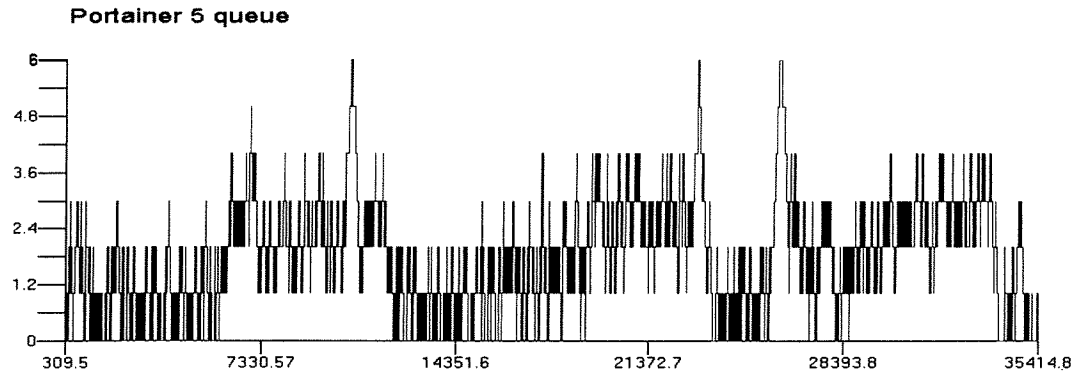


Figure 25.5, queue under portainer 5 while unloading a mega vessel

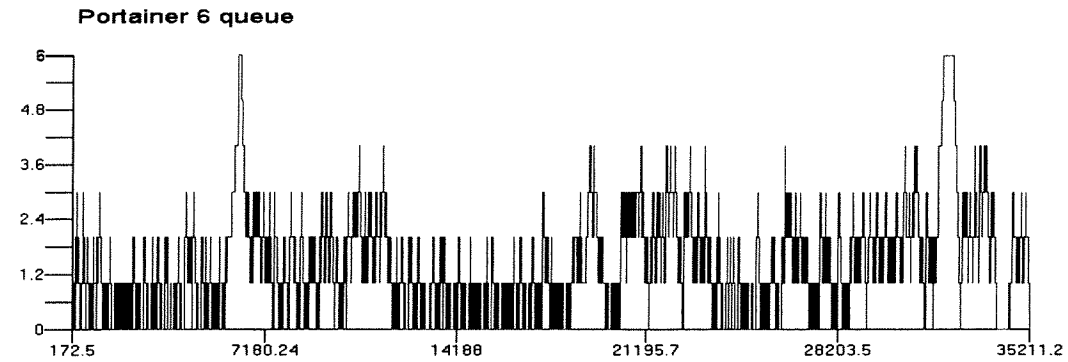


Figure 25.6, queue under portainer 6 while unloading a mega vessel

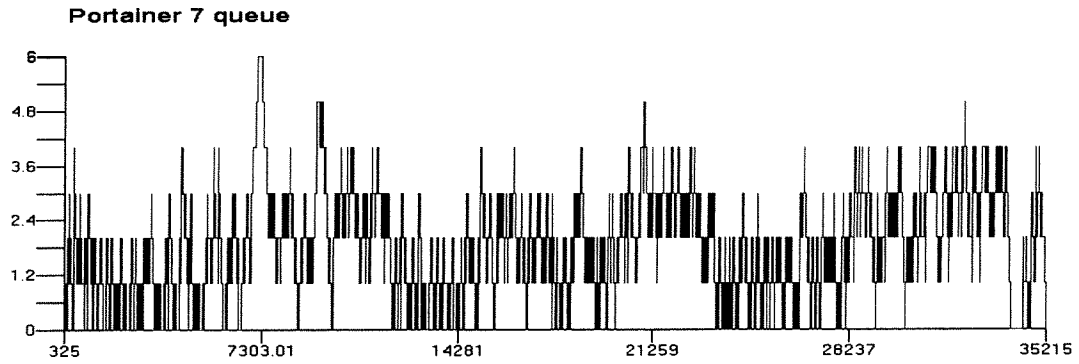


Figure 25.7, queue under portainer 7 while unloading a mega vessel

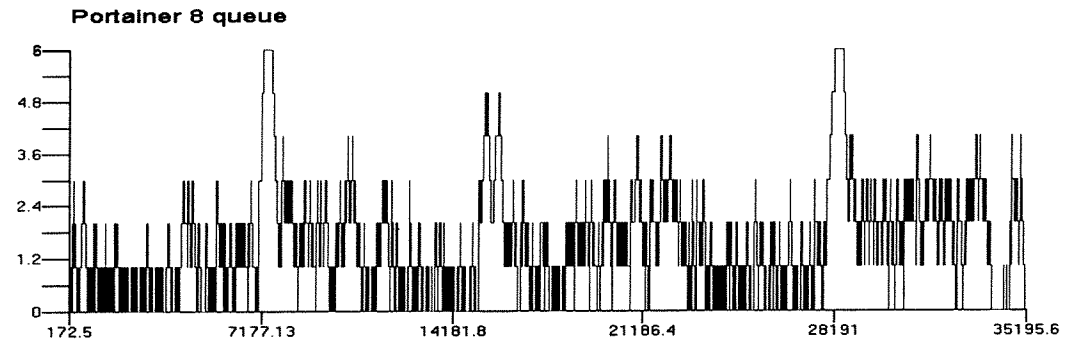


Figure 25.8, queue under portainer 8 while unloading a mega vessel

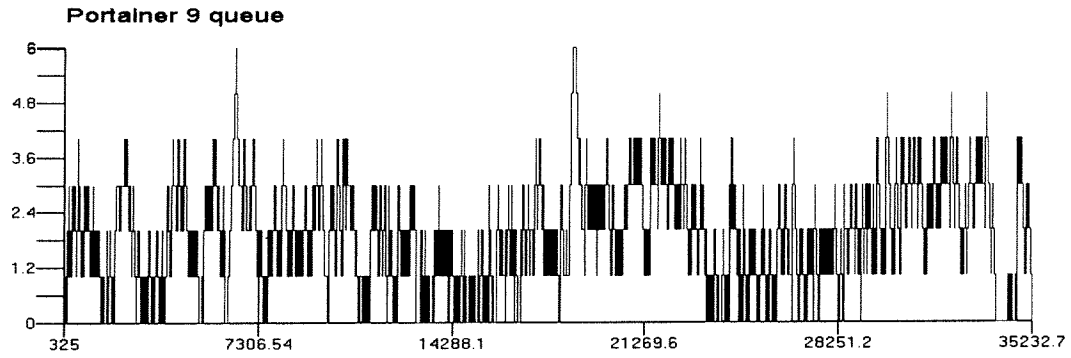


Figure 25.9, queue under portainer 9 while unloading a mega vessel

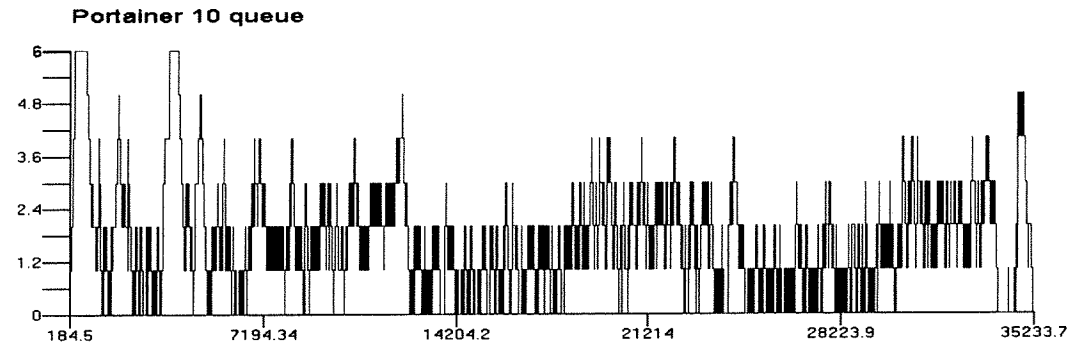


Figure 25.10, queue under portainer 10 while unloading a mega vessel

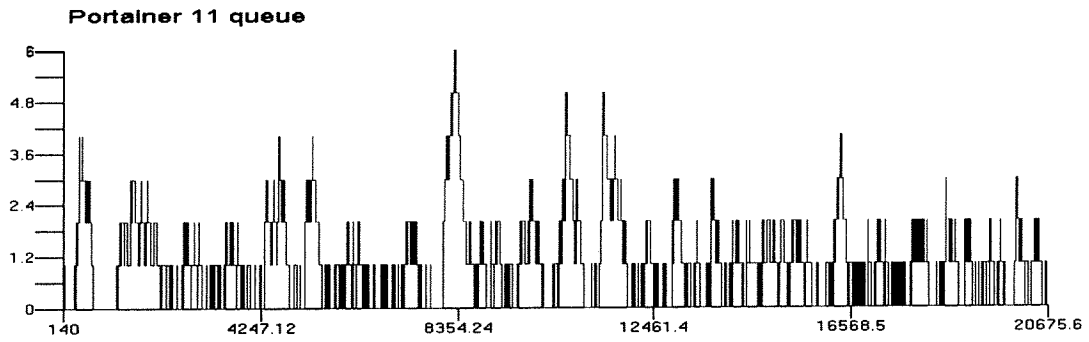


Figure 25.11, queue under portainer 11 while loading a feeder from the adjacent berth

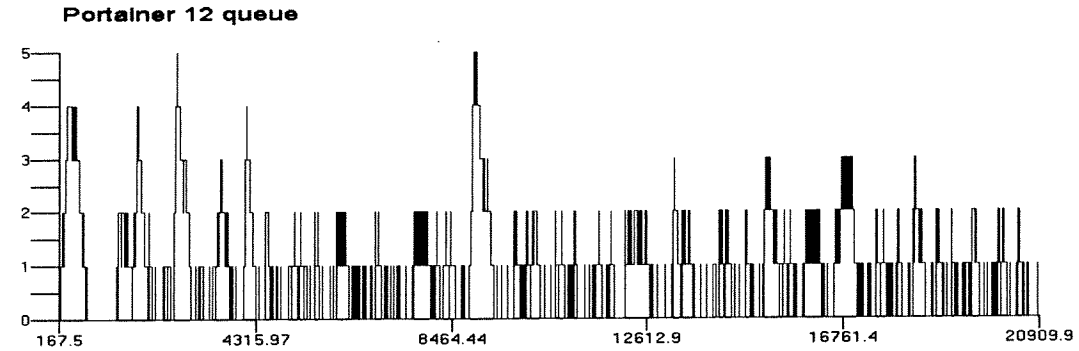


Figure 25.12, queue under portainer 12 while loading a feeder from the adjacent berth

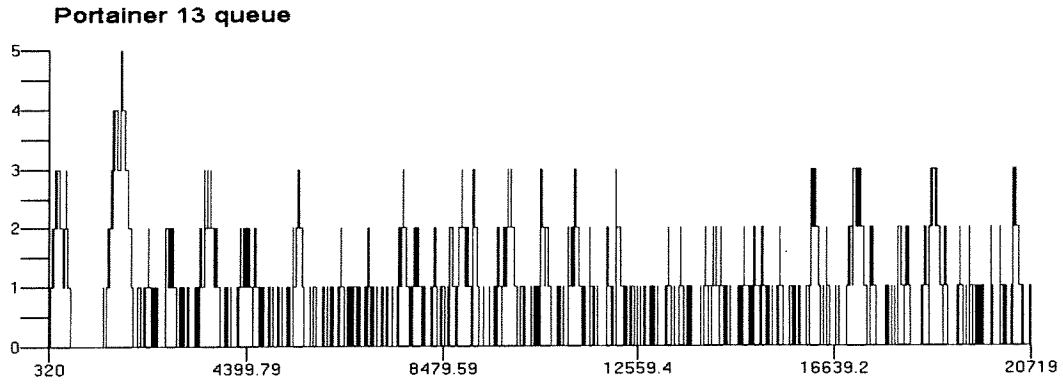


Figure 25.13, queue under portainer 13 while loading a feeder from the adjacent berth

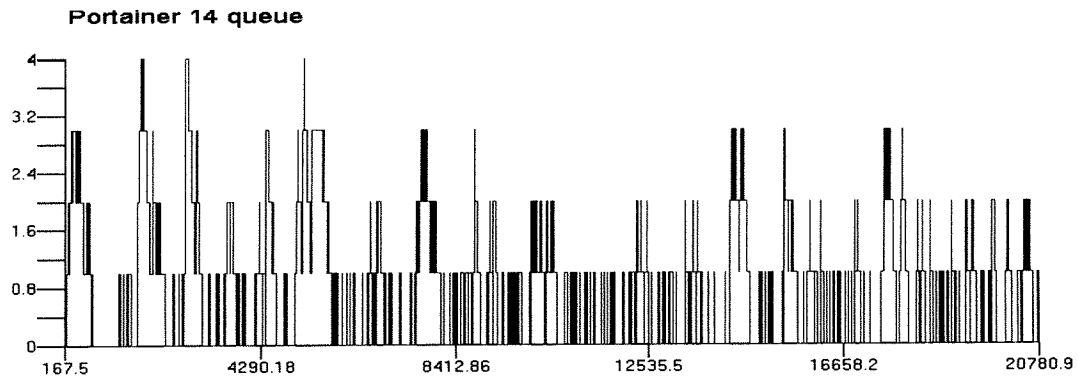


Figure 25.14, queue under portainer 14 while loading a feeder from the adjacent berth

Appendix 26: Input parameters for the experiment with variable crane cycle times.

The basic parameters used for this experiment are shown in Table 26.1. The parameters are the same as used in experiment 10, except for the crane cycle times. The cycle times in this experiment can differ from 44 up to 116 seconds, with a mean cycle time of 80 seconds.

Parameter	Value	
frombtob[1,2]	0	containers
frombtob[3,2]	0	containers
frombtob[2,1]	1600	containers
frombtob[2,3]	1600	containers
inplacecounter 1	0	containers
inplacecounter 2	0	containers
inplacecounter 3	0	containers
seed of uniform	67	-
seed of arrivaltimes feeders	17	-
seed of arrivaltimes big	27	-
import generator feeders	40	containers
export generator feeders	1050	containers
import generator big vessels	3150	containers
export generator big vessels	0	containers
rgv full speed	3	m/s
rgv low speed passing a shortcut	2	m/s
rgv node speed passing a node	1	m/s
rgv acceleration	1	m/s^2
minimum distance between trolleys	3	m
number of stacking cranes at terminal	90	-
number of agv's per portainer used:		
- unloading a vessel	6	-
- loading a vessel from different berth	10	-
- shuffling at the terminal	10	-
mean stacking crane cycle time	**80**	seconds
mean portainer cycle time	58	seconds

Table 26.1 input parameters for experiment; Adjacent loading at Berth 1 and Berth 3 and unloading at Berth 2, using variable crane cycle times.

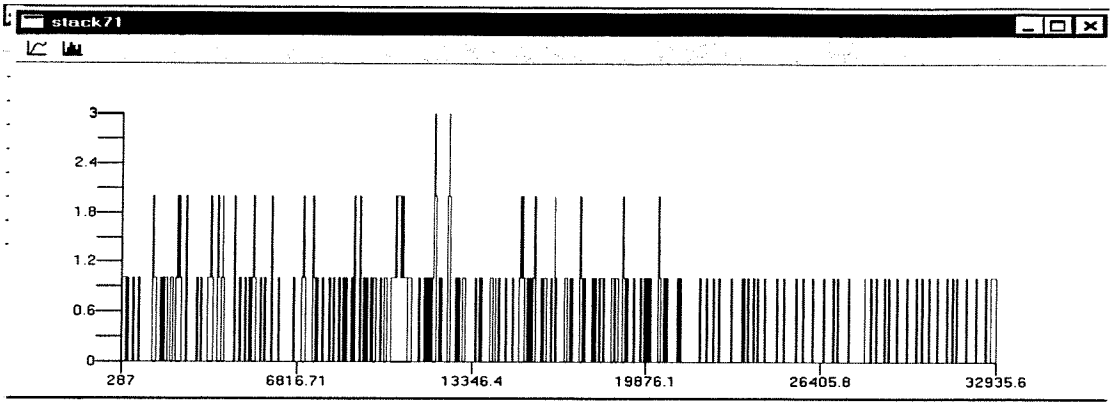


Figure 26.1, rgv queue on warehouse lane 71

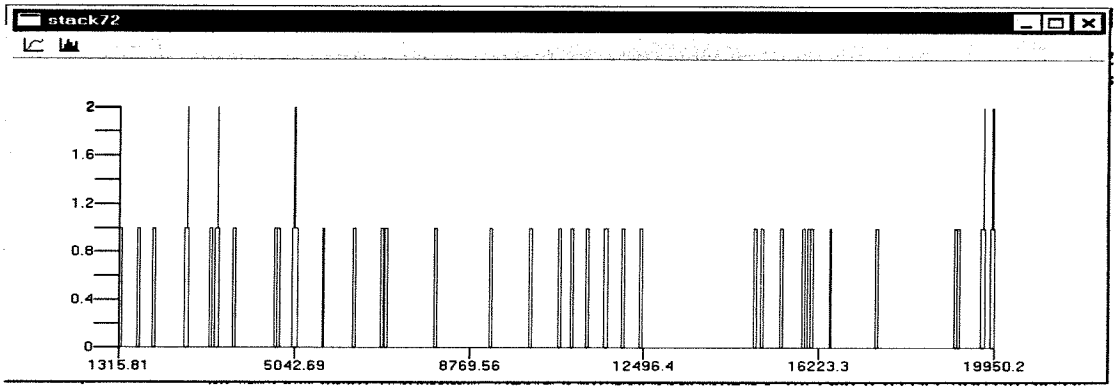


Figure 26.2, rgv queue on warehouse lane 72

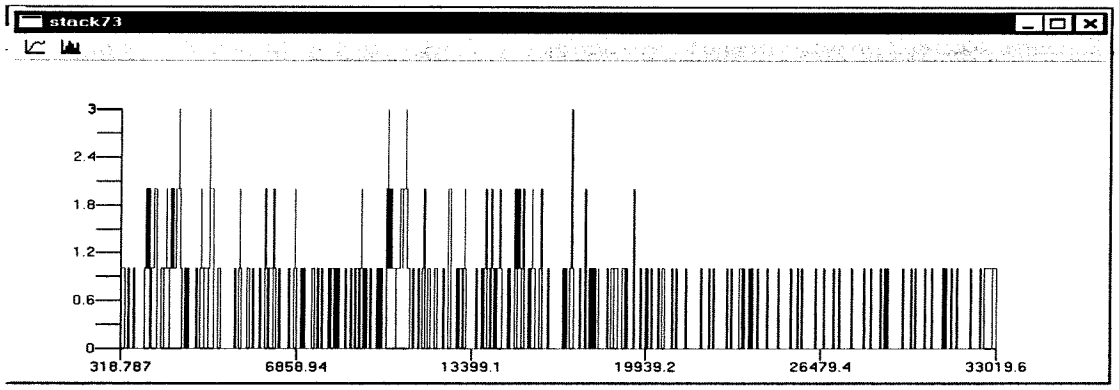


Figure 26.3, rgv queue on warehouse lane 73

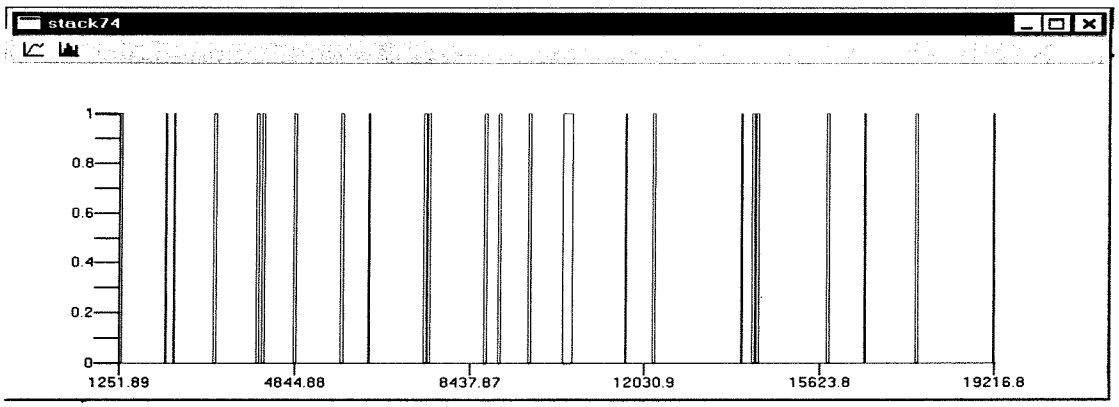


Figure 26.4, rgv queue on warehouse lane 74

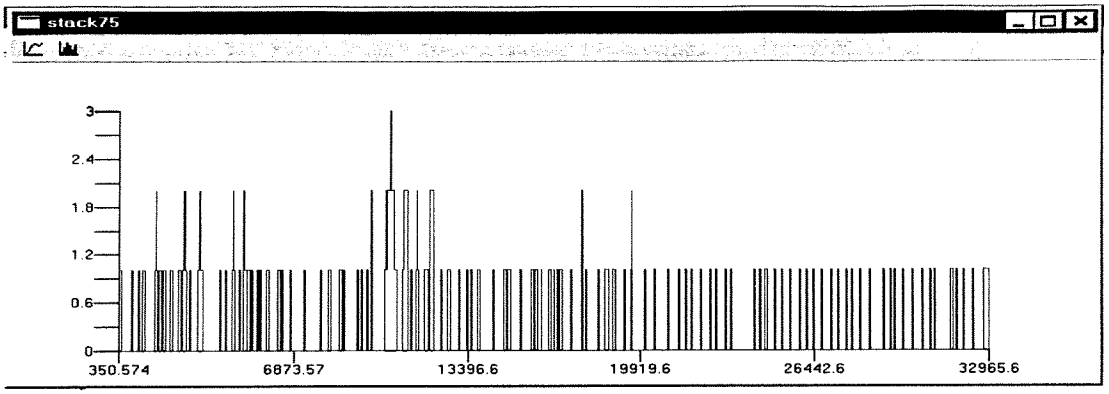


Figure 26.5, rgv queue on warehouse lane 75

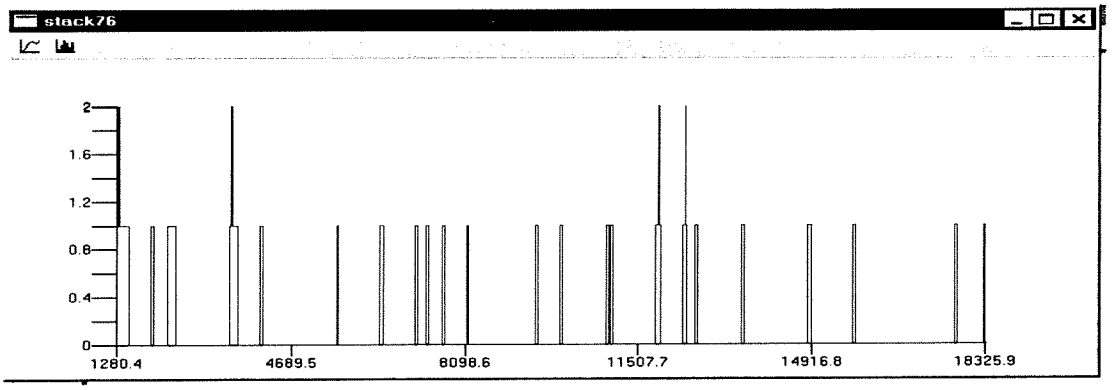


Figure 26.6, rgv queue on warehouse lane 76

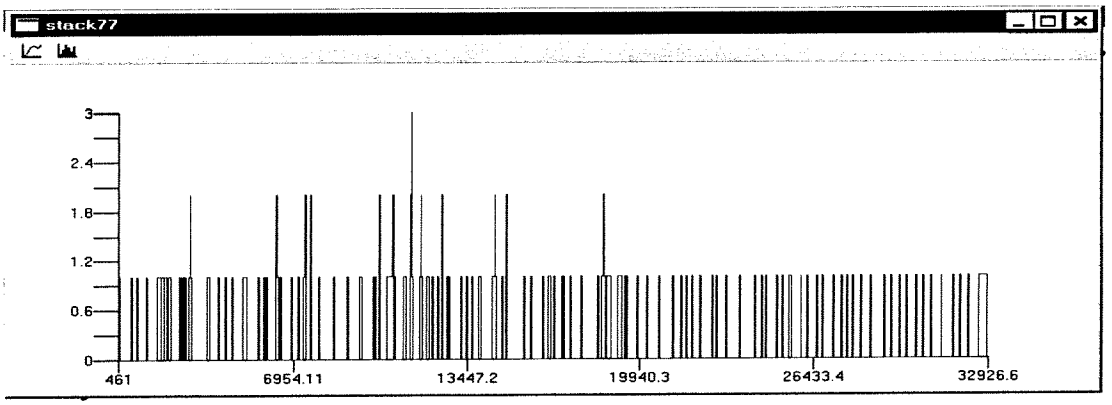


Figure 26.7, rgv queue on warehouse lane 77

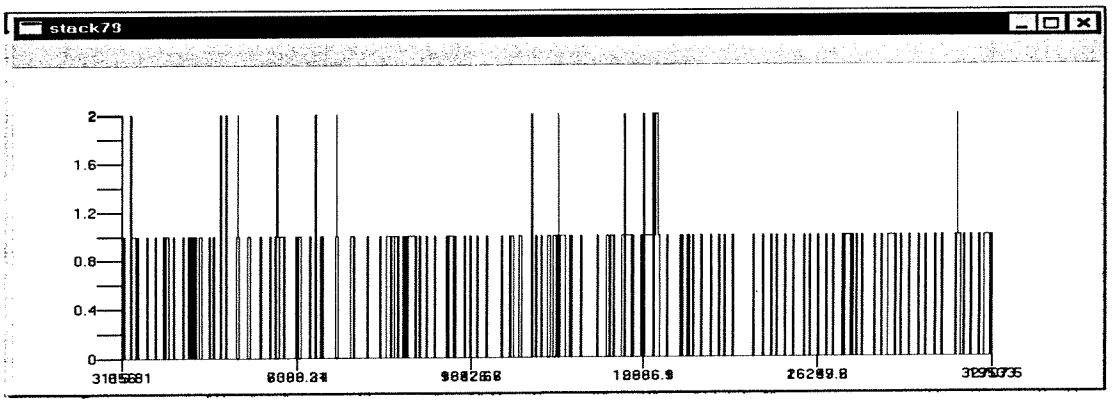


Figure 26.8, rgv queue on warehouse lane 79

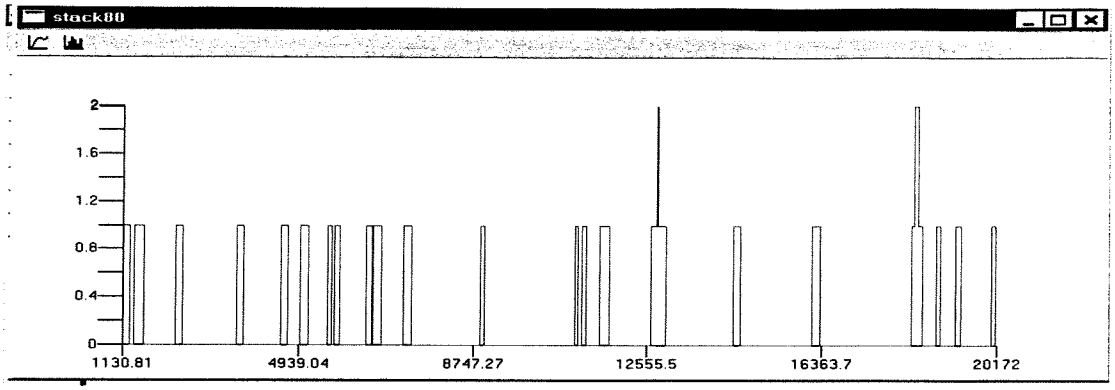


Figure 26.9, rgv queue on warehouse lane 80

Appendix 27 Table section

AVERAGE WAITING TIME OF CUSTOMERS IN THE QUEUE M/D/n (IN UNITS OF AVERAGE WAITING TIME)

utilization	Number of Servers (n)								
(u)	2	3	4	5	6	7	8	9	10
0.1	0.0062	0.0009	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.2	0.0242	0.0066	0.0021	0.0007	0.0002	0.0001	0.0000	0.0000	0.0000
0.3	0.0553	0.0201	0.0085	0.0039	0.0019	0.0009	0.0005	0.0002	0.0001
0.4	0.1033	0.0450	0.0227	0.0124	0.0072	0.0043	0.0026	0.0017	0.0011
0.5	0.1767	0.0872	0.0497	0.0307	0.0199	0.0135	0.0093	0.0066	0.0047
0.6	0.2930	0.1584	0.0984	0.0661	0.0467	0.0342	0.0257	0.0197	0.0154
0.7	0.4936	0.2862	0.1897	0.1355	0.1016	0.0788	0.0627	0.0508	0.0419
0.8	0.9030	0.5537	0.3860	0.2890	0.2265	0.1833	0.1519	0.1282	0.1098
0.9	2.0138	1.2887	0.9340	0.7237	0.5848	0.4894	0.4164	0.3606	0.3175

Table I

AVERAGE WAITING TIME OF CUSTOMERS IN THE QUEUE D/M/n (IN UNITS OF AVERAGE SERVICE TIME)

utilization	Number of Servers (n)								
(u)	2	3	4	5	6	7	8	9	10
0.1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.2	0.0003	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.3	0.0048	0.0008	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.4	0.0223	0.0060	0.0019	0.0007	0.0002	0.0001	0.0000	0.0000	0.0000
0.5	0.0649	0.0239	0.0103	0.0049	0.0024	0.0013	0.0007	0.0004	0.0002
0.6	0.1520	0.0685	0.0360	0.0206	0.0125	0.0079	0.0051	0.0034	0.0023
0.7	0.3257	0.1696	0.1020	0.0665	0.0458	0.0327	0.0240	0.0180	0.0137
0.8	0.7111	0.4114	0.2725	0.1947	0.1461	0.1134	0.0903	0.0734	0.0605
0.9	1.9330	1.2112	0.8612	0.6567	0.5238	0.4310	0.3629	0.3110	0.2703

Table II

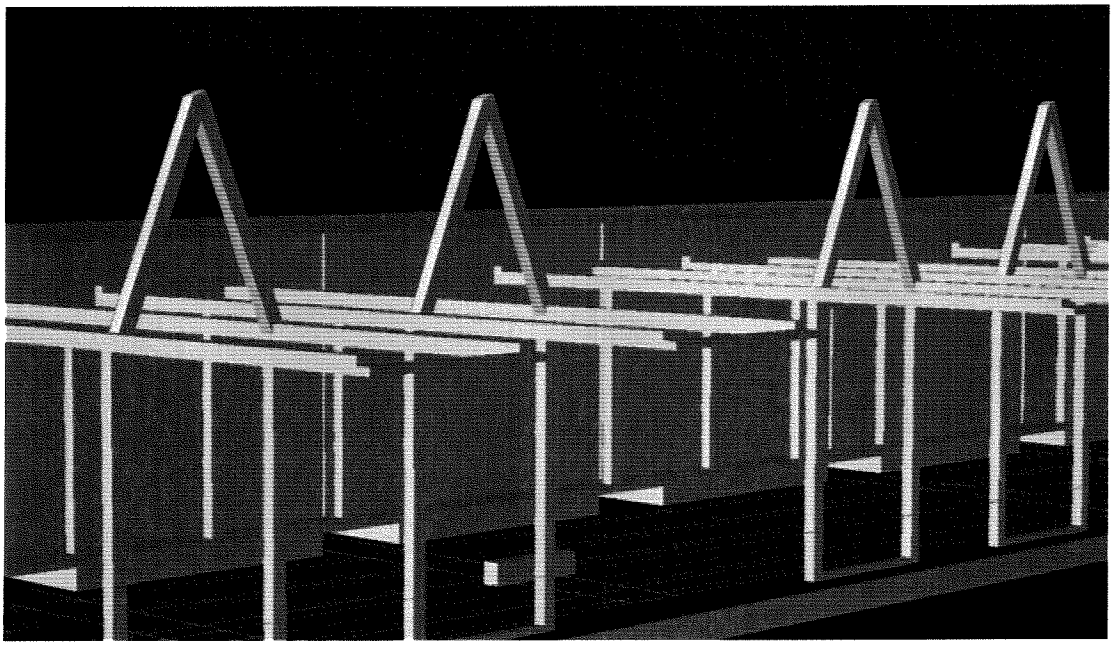
AVERAGE WAITING TIME, IN UNITS OF AVERAGE SERVICE TIME, FOR THE SINGLE SERVER QUEUE $E_k/E_l/1$ WITH UTILISATION $u = 0.6$

	Value of 1 for the Erlang Service Distribution = $1/v$,									
$s_e = 1/k$	1	2	3	4	5	6	7	8	9	10
0.1	0.5786	0.2600	0.1643	0.1200	0.0951	0.0793	0.0684	0.0606	0.0547	0.0501
0.2	0.6786	0.3488	0.2463	0.1973	0.1689	0.1504	0.1375	0.1280	0.1207	0.1149
0.3	0.7996	0.4407	0.3330	0.2808	0.2500	0.2299	0.2156	0.2050	0.1968	0.1903
0.4	0.8813	0.5348	0.4232	0.3684	0.3360	0.3146	0.2994	0.2880	0.2793	0.2723
0.5	0.9835	0.6306	0.5157	0.4590	0.4253	0.4029	0.3870	0.3751	0.3659	0.3586
0.6	1.0862	0.7278	0.6102	0.5519	0.5171	0.4940	0.4775	0.4652	0.4556	0.4480
0.7	1.1892	0.8259	0.7061	0.6465	0.6108	0.5871	0.5702	0.5576	0.5477	0.5399
0.8	1.2926	0.9249	0.8031	0.7424	0.7061	0.6819	0.6646	0.6516	0.6416	0.6335
0.9	1.3962	1.0247	0.9012	0.8395	0.8025	0.7778	0.7603	0.7471	0.7369	0.7287
1.0	1.5000	1.1250	1.0000	0.9375	0.9000	0.8750	0.8571	0.8438	0.8333	0.8250

Table III

Study on the Designing and Simulation of a High capacity transshipment terminal

- ***Appendix 28 Listing***



Impression of the portainers and container warehouse

December 2000
Hein van Hees
Thies Visscher

TUDelft
Delft University of Technology

Faculty of Civil Engineering and Geosciences
Department of Hydraulic and Geotechnical Engineering
Section of Hydraulic Engineering

28-I. Define section

28-II. Modules

28-III. Macro's

28-IV. Clarifying word list

28-V. Input file for lay out of the terminal

28-VI Input file Rgv attributes

Appendix 28-I Define section

```
DEFINE
     COMPONENT
          MAIN
               INTEGER
                    N
                    I
                    SHUFFLECOUNTER
                    A
                    BERTHNR
                    FROMBTOB[ 3,3]
                    Q
                    J
                    N_NODES
                    SUPERNODE
                    PTL
                    INPLACECOUNTER[ 3]
               LOGICAL
                    QUAYLOCATION[ 18]
               REAL
                    K
                    SHIPTURNAROUNDTIME
                    N_IN
                    N_OUT
                    ACC
                    FULL_SPEED
                    LOW_SPEED
                    NODE_SPEED
                    HANDLING
                    DISTANCE
                    MN
                    VBNM
                    MNBV
                    NM
                    MNB
                    BMN
                    C1
                    C2
                    C3
                    C4
                    C5
                    C6
                    C7
                    C8
                    C9
                    C10
               CHARACTER
                    SHIPATIME[ 12]
                    PLACES[ 4]
                    OUTSTRING[ 14]
                    CHAR[ 22]
                    CHAR9[ 9]
               REFERENCE TO DATASTREAM
                    OUTPORT
                    OUTPORTU
                    HCONFIG
                    OUTPORTATION[ 300]
               REFERENCE TO POINTSTREAM
                    SDWELLTIMES
                    AGVCYCLETIMES
                    OUT
                    SHIPTURNAROUNDTIMES
                    DOORLOOP
               REFERENCE TO DISTRIBUTION
```

ARRIVALTIMES
UNIF
ANIF
ARRIVALTIMESBIG
REFERENCE TO WINDOW
AGVCYCLTIMES
SHIPDWELLTIMES
SHTURNAROUNDTIMES
CHART
DOORWIN
ANCHORAGETIMES
PORTAINER1QUEUE
BERTHOCCUPIED[3]
PORTAINERSACTIVE
PORT18QUEUE
PORT17QUEUE
PORT16QUEUE
PORT15QUEUE
PORT14QUEUE
PORT13QUEUE
PORT12QUEUE
PORT11QUEUE
PORT10QUEUE
PORT9QUEUE
PORT8QUEUE
PORT7QUEUE
PORT6QUEUE
PORT5QUEUE
PORT4QUEUE
PORT3QUEUE
PORT2QUEUE
PORT1QUEUE
CONTROL
CRANE2QUEUE
SECROW[200]
REFERENCE TO WOB
CONS
B_OK
PRM_MT
PRM_M
PRM_ST
PRM_MN
STR_MT
STR_M
STR_MP
STR_MN
PRM_MAXN
STR_TEMP
STR_MAXN
PRM_MP
PRM_TEMP
STR_01
STR_02
STR_03
STR_04
STR_05
STR_06
STR_07
STR_08
STR_09
STR_10
PRM_10
PRM_01

```
          PRM_02
          PRM_03
          PRM_04
          PRM_05
          PRM_06
          PRM_07
          PRM_08
          PRM_09
     REFERENCE TO SET
          FROMPARKLIST
          CLAIMSET
          ANCHORAGE
          QUAYRAIL
          BERTH[ 3]
          AGVPARKING
          CRANELIST
          AGVWORKLIST
          ACTIVEAGVLIST
          LOADINGSHIP
          UNLOADINGSHIP
          PORTLOCATIONS
          ACTIVESHIPLIST
          QUAYMWORKLIST
          BERTHAGVWAITLIST
          PORTAINERACTIVELIST
          PLATFORMS
          SECTIONS
          STACKLIST
          NODE_LIST
          AGVBERTHSHUFFLELIST[ 3]
          PORTAINERPERBERTH[ 3]
          GONESHIPLIST
     REFERENCE TO PORTAINER
          NEWPORTAINER
     REFERENCE TO AGV
          NEWAGV
     REFERENCE TO CRANE
          NEWCRANE
     REFERENCE TO PORTLOC
          NEWPORTLOC
     REFERENCE TO STACK
          NEWSTACK
          STACKS[ 40]
     REFERENCE TO NODE
          NODES[ 800]
     REFERENCE TO SECTION
          NEWSECTION
SHIPGEN
     INTEGER
          S
          R
     REAL
          TIMEBIG
          TIMESMALL
     REFERENCE TO SHIP
          NEWSHIP
BERTHMASTER
     LOGICAL
          OCCUPIED[ 3]
     REFERENCE TO SHIP
          BERTHMASTERSHIP
PORTAINERMASTER
     INTEGER
```

```
          E
          ACTIVEPORT
     REAL
          TIME
     REFERENCE TO SHIP
          PTMSHIP
     REFERENCE TO PORTAINER
          PORTAINERMASTERPORTAINER
     REFERENCE TO PORTLOC
          PMPORTLOC
IMPORTGENERATOR
     REAL
          WW
     REFERENCE TO SHIP
          IMPORTSHIP
EXPORTGENERATOR
     REFERENCE TO SHIP
          EXPORTSHIP
AGVMASTER
     INTEGER
          PAGV
          PASSAGV
          ACTAGV
          ACTIVEAGV
          ASL
          NEEDEDAGVS
          PAL
          RI
          RA
          RO
          RU
     REFERENCE TO SHIP
          AMSHIP
     REFERENCE TO PORTAINER
          AGVPORTAINER
     REFERENCE TO AGV
          BYEAGV
          HIAGV
          DEBYEAGV
TRAFFICMASTER
     INTEGER
          V4
          V3
          V2
          V1
          WHERE
          TMX
          TMSHIPNUMBER
          TMTOTAL
          TMTOTALIM
          TMTOTALEX
          TMY
          TMYY
          TMXX
     REAL
          LASTK
          CYCLETIME
          TMRES40IM
          TMRES40EX
          W
          TMRESTOTEX
          W1
          W2
```

```
                W3
                TMRESTOTIM
                W5
                W4
                W6
          REFERENCE TO SHIP
                TMSHIP
          REFERENCE TO PORTAINER
                TMPORT
          REFERENCE TO AGV
                TMAGV
          REFERENCE TO CRANE
                TMCRANE
          REFERENCE TO PORTLOC
                TMPORTLOC
          REFERENCE TO STACK
                GOEDE_STACK
          REFERENCE TO NODE
                GOEDE_NODE
          REFERENCE TO SECTION
                GOEDE_SECTION
CLASS
     SHIP
          INTEGER
                FIRSTSHUFFLEAGV
                SHIPBERTHNUMBER
                SHIPNUMBER
                TOTALLOAD
                TOTALUNLOAD
                IM20
                EX40
                IM40
                EX20
                TOTAL
                TEX
                TOTALU
          REAL
                SHIPDWELL
                SHIPANCHORAGETIME
                SHIPARRIVALTIME
                PROPEX20
                PROPIM40
                PROPEX40
                PROPIM20
     PORTAINER
          INTEGER
                PORTAINERNUMBER
                C
                PTBERTHNR
                T
                H
                PORTAINERSHIPNB
                FUTPORTAINERSHIPNB
                ACTLOCNR actual location number
                FUTLOCNR future location number
                L
                PP
                AGVPPNEEDED
                REP
                PERS
                PER
                TT
          REAL
```

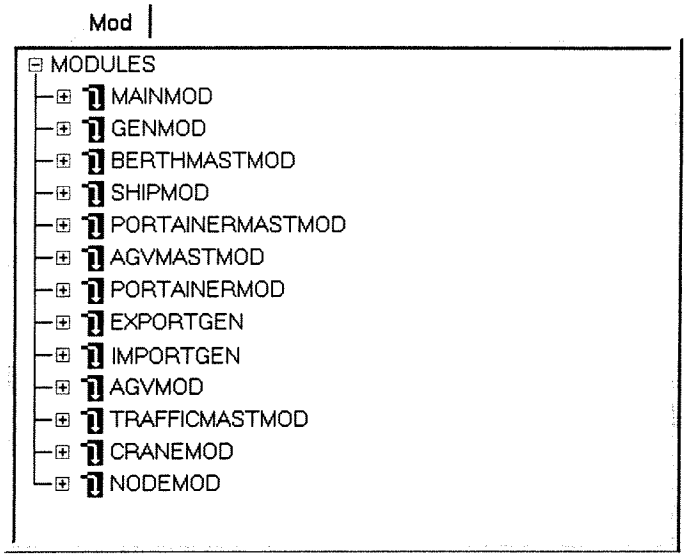
```
          QQ
     REFERENCE TO SET
          PORTQUEUE
          PORTASQUEUE
          AGVPPACTIVE_ODD
          AGVPPACTIVE
          AGVPPACTIVER
     REFERENCE TO SHIP
          PTSHIP
     REFERENCE TO AGV
          PORTBYEAGV
          ASAGV
          PORTAGV
     REFERENCE TO PORTLOC
          PTPORTLOC
AGV
     INTEGER
          NR
          SBNR
          AGVNUMBER
          XX
          Y
          AGVLOC
          AGVBERTHNB
          YY
          LE
          X
          LL
          YYY
          XXX
          ZZZ
          RR
          BB
     LOGICAL
          CONT20SHUF
          CONT20FT
          WEIRD
          EASYLOAD
          LOADEDAGV
          YARD
          BYE
          FIRSTSHUFFLE
          DELAYED
          Z
          AGVBIG
          AGVRED
          AGVGREEN
          AGVBLUE
          SHUFFLEAGV
          UNEASY
          BEENINSTACK
     REAL
          FREE_WAY
          CLAIMEDLENGTH
          REACHABLE_POINT
          B
          ARRIVTIME
     REFERENCE TO SET
          ROUTE
          ENTRANCE
          SUPERROUTE
     REFERENCE TO AGV
          HEADER
```

```
     REFERENCE TO CRANE
          AGVCRANE
     REFERENCE TO PORTLOC
          AGVPORTLOC
     REFERENCE TO STACK
          AGVSTACK
     REFERENCE TO NODE
          EXIT
          SUPEREXIT
          ENTRANCE_NODE
          BEFORE_ENTRANCE_NODE
          REFERENCE_NODE
          CLAIM_NODE
          CLAIM2
          CLAIM1
          LASTNODE
          EXEPT_NODE
          CUR_NODE
     REFERENCE TO SECTION
          CLAIM_SEC
          CUR_SEC
          AGV_SECTION
     REFERENCE TO PLATFORM
          STOP
CRANE
     INTEGER
          CRANECLUSTERNR
          COUNTER
          CRANENUMBER
          AANTALAGV
          READYCNT
          KLAARCOUNT
     LOGICAL
          SLEEP
     REAL
          CONTAINERS
          F
          G
     CHARACTER
          CRANENAME[ 8]
     REFERENCE TO SET
          CRANEQUEUE
          READYQUEUE
          CRANELOSSENQUEUE
          CRANELADENQUEUE
          ASSIGNEDQUEUE
     REFERENCE TO AGV
          CRANEAGV
          ASSIGNEDAGV
     REFERENCE TO STACK
          CRANESTACK
PORTLOC
     INTEGER
          PORTLOCNUMBER
          TELLER40IM
          TELLER20IM
     REFERENCE TO SET
          PORTLOCQUEUE
          LOCATIONSET
          ASSIGNEDAGVS
STACK
     INTEGER
          STACKNUMBER
```

```
          P
          D
     REAL
          ST_X
          ST_Y
          STACKCOUNT
          BGCOUNT
          SGCOUNT
          BRCOUNT
          SRCOUNT
          BBCOUNT
          SBCOUNT
     REFERENCE TO SET
          WANTASQUEUE
          STACK_ROW
     REFERENCE TO AGV
          STACKAGV
     REFERENCE TO CRANE
          THECRANE
     REFERENCE TO NODE
          IN_STACK_NODE
     REFERENCE TO SECTION
          WAY_IN_STACK
          WAY_OUT_STACK
NODE
     INTEGER
          CLAIMED
          NODEAGVNB
          NODE_NUMBER
     REAL
          ND_X
          ND_Y
          CLAIMTIME
          EXITTIME
          NODESPEED
          NODEDISTANCE1
          NODEDISTANCE2
     MACRO
          SIDE_TRACK[ 2]
     REFERENCE TO AGV
          NODEAGV
     REFERENCE TO SECTION
          OUTS[ 3]
SECTION
     REAL
          S_LENGTH
          SEC_ROW_NUMBER
     REFERENCE TO SET
          SEC_ROW
     REFERENCE
          END_POINT
PLATFORM
     REAL
          PL_X
          PL_Y
     REFERENCE TO POINTSTREAM
          INTERVAL_CHECK
     REFERENCE TO SET
          ROW
     REFERENCE TO AGV
          AGVPLATFORM
     REFERENCE TO NODE
          IN_NODE
```

```
          REFERENCE TO SECTION
               WAY_IN
               WAY_OUT
EXTERNAL
TIMEUNIT
     SECOND
```

Appendix 28-II Modules



The main starts with the creating of different sets.

Node_list is a set containing all the nodes

Agvberthshufflelist [1 to 3] are 3 sets containing all the rgv's shuffling from that berth

Activeshiplist a set containing the active vessels
Stacklist a set containing all the stacks
Agvparking a set containing the not active rgv's
Agvworklist a set containing rgv's with work requests for the trafficmaster

Cranelist a set containing all the cranes
Loadingship a set containing all the loading vessels at the terminal

Outportation creates the output files
Goneshiplist a set containing the terminated vessels, in this way their attributes can be stored

Portainerperberth [1 to 3] keeps count of the active portainers per berth

Portaineractivelist a set containing all the active portainers
Portlocations a set containing all the portainerlocations

Quayrail a set containing all the portainers

Occupied, is a logical if a berth is taken by a vessel occupied becomes true

Unloadingship a set containing all the vessels that are unloading

In **line 57 to 72** the portainers are created

n = number of portainers that have to be created

```
 1 @set creation@
 2 r←0
 3 doorloop ← NEW POINTSTREAM
 4 node_list ← NEW SET
 5 activeagvlist ← NEW SET WITH HISTORY
 6 agvberthshufflelist[ 1] ← NEW SET
 7 agvberthshufflelist[ 2] ← NEW SET
 8 agvberthshufflelist[ 3] ← NEW SET
 9 activeshiplist ← NEW SET
10 stacklist ← NEW SET
11 agvparking ← NEW SET WITH HISTORY
12 agvworklist ← NEW SET
13 claimset ← NEW SET
14 agvcycletimes ← NEW POINTSTREAM
15 anchorage ← NEW SET WITH HISTORY
16 berth[ 1] ← NEW SET WITH HISTORY
17 berth[ 2] ← NEW SET WITH HISTORY
18 berth[ 3] ← NEW SET WITH HISTORY
19 berthagvwaitlist ← NEW SET
20 cranelist ← NEW SET
21 loadingship ← NEW SET
22 outport ← NEW DATASTREAM
23 outportu ← NEW DATASTREAM
24 geluk ← NEW POINTSTREAM
25 out ← NEW POINTSTREAM
26 JUMP TO create_outportation
27 goneshiplist←NEW SET
28 ATTACH "shiparrivaltimes.txt" TO outportu
29 ATTACH "concout.txt" TO outport
30 portainerperberth[ 3] ← NEW SET WITH HISTORY
31 portainerperberth[ 2] ← NEW SET WITH HISTORY
32 portainerperberth[ 1] ← NEW SET WITH HISTORY
33 portaineractivelist ← NEW SET WITH HISTORY
34 portlocations ← NEW SET
35
36 geluk ← NEW POINTSTREAM
37 out ← NEW POINTSTREAM
38
39
40
41 quayrail ← NEW SET
42 quaymworklist ← NEW SET
43
44 occupied[ 1] ← FALSE
45 occupied[ 2] ← FALSE
46 occupied[ 3] ← FALSE
47
48 unloadingship ← NEW SET
49
50 sdwelltimes ← NEW POINTSTREAM
51 shipturnaroundtimes ← NEW POINTSTREAM
52
53 k ← 0.225
54 s ← 1
55
56
57 @@@@@@@@@@@@@@@@@@@@@@@@@@@
58 @creating the portainers:@
59 @@@@@@@@@@@@@@@@@@@@@@@@@@@
60
61 n ← 18@10
62 FOR i ← 1 TO n
```

actlocnr is the actual location number, the location where the portainer is at that moment

agvppactive_odd is a set containing the number of odd agv's per portainer active

agvppactive is a set containing the number of agv's (rgv's) per portainer active

From **line 74 to 95** portainer locations are created

n is the number of portainerlocations to be created

portloc is short for portainerlocation

quaylocation[i] is true means this location is taken by a portainer

From line **96 to 112** the rgv's and their attributes are created

n is the number of rgv's that have to be created

Route of the rgv is a set containing all the exits of the agv

Entrance of the rgv is a set containing the starting point of a route of the rgv

x refers to the current portlocation of the rgv
y refers to the current stack of the rgv

Loadedagv is a logical, false means the rgv is not carrying a container

From **line 115 to 150** the cranes are created

n is the number of cranes that have to be created

```
 63      newportainer ← NEW portainer
 64      portainernumber OF newportainer ← i
 65      actlocnr OF newportainer ← i
 66      portqueue OF newportainer ← NEW SET WITH HISTORY
 67      portasqueue OF newportainer ← NEW SET
 68      agvppactive_odd OF newportainer←NEW SET
 69      agvppactive OF newportainer ← NEW SET
 70      agvppactiver OF newportainer ← NEW SET
 71      JOIN newportainer TO quayrail
 72  END
 73
 74  @@@@
```

cranequeue is a set containing the rgv's waiting to be served by the crane
assignedqueue is a set containing rgv's assigned to the crane. When a rgv is assigned
the crane can start working if possible

cranelist a set containing all the cranes

frombtob[1,2] is 4000 tells there are 4000 containers in the stacks behind berth 1
which are destined to be taken away by a vessel from berth 2

Inplacecounter[1] counts how many containers are destined to be taken away
from that berth by a vessel arriving at that berth. The containers can be loaded easy
when a vessel arrives at that berth.

The feeders are generated

The mega vessels are created

From **line 178 to 199** the input parameters are imported from a file
This file is called hconvig

acc is the acceleration of the rgv's to be used during the run

```
125     counter OF newcrane ← 0
126     aantalagv OF newcrane←0
127     klaarcount OF newcrane ← 0
128     IF ((i-1)MODULO(10))+1 ≤ 4
129         cranename OF newcrane ← "40"
130         craneclusternr OF newcrane ← ptl
131     END
132     IF ((i-1)MODULO(10))+1 > 4 & ((i-1)MODULO(10))+1 ≤ 9
133         cranename OF newcrane ← "20"
134         craneclusternr OF newcrane ← ptl
135     END
136     IF ((i-1)MODULO(10))+1 = 10
137         cranename OF newcrane ← "20"
138         craneclusternr OF newcrane ← ptl
139
140         ptl ← ptl + 1
141     END
142
143     cranequeue OF newcrane ← NEW SET
144     assignedqueue OF newcrane ← NEW SET
145     cranelossenqueue OF newcrane ← NEW SET
146     craneladenqueue OF newcrane← NEW SET
147     readyqueue OF newcrane ← NEW SET
148     containers OF  newcrane ← 10
149     JOIN newcrane TO cranelist
150 END
151
152 ACTIVATE EACH crane IN cranelist FROM begin IN cranemod
153
154 shufflecounter←0
155 frombtob[ 1,2] ←4000
156 frombtob[ 3,2] ←4000
157 frombtob[ 2,1] ←4000
158 frombtob[ 2,3] ←4000
159 inplacecounter[ 1] ←4000
160 inplacecounter[ 2] ←000
161 inplacecounter[ 3] ←000
162
163 unif ← NEW DISTRIBUTION
164 ATTACH UNIFORM(0,1) TO unif
165 SEED OF unif ← 67
166
167 @Feeders@
168 arrivaltimes ← NEW DISTRIBUTION
169 ATTACH  NORMAL(10,0) TO arrivaltimes
170 SEED OF arrivaltimes ← 17
171
172 @Mega Vessels@
173 arrivaltimesbig ← NEW DISTRIBUTION
174 ATTACH  NORMAL(15,0) TO arrivaltimesbig
175 SEED OF arrivaltimes ← 27
176
177
178 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
179 @ creating the streetplan     @
180 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
181
182 hconfig ← NEW DATASTREAM
183 ATTACH READ FROM CASE TO hconfig
184 OPEN hconfig FOR INPUT
185
186 acc         ← READ FROM CASE
```

full_speed is the top speed of the rgv, this speed can be reached on the main tracks
nodespeed is the speed of a rgv when it crosses a node
low_speed is the speed of the rgv in the stacks and at the short cuts

distance is the minimum distance between rgv's when driving on the terminal

From **line 200 to 272** the main tracks of the terminal are created

In **line 206,** 800 nodes are created, these shape the framework of the terminal

nd_x is the x co-ordinate of node 1
nd_y is the y coordinate of node 1

between nodes sections can be created

end_point is a reference to the last node of a section
s_length is the length of section calculated by the function distance to

sections is a set containing all the sections

different section require different driving speeds
Node 1 to 118 and 501 to 618 create main routes, the driving speed at the main route is full_speed

node_distance refers to the safety distance at the tracks, this can differ per track

```
187 full_speed  ← READ FROM CASE
188 node_speed  ← READ FROM CASE
189 low_speed   ← READ FROM CASE
190 acc         ← acc@*3600
191 full_speed  ← full_speed @*60
192 node_speed  ← node_speed @*60
193 low_speed   ← low_speed @*60
194 handling    ← READ FROM CASE
195 distance    ← READ FROM CASE
196
197 sections  ← NEW SET CALLED "sections"
198 platforms ← NEW SET CALLED "platforms"
199
200 @@@@@@@@@@@@@@@@
201 @  main track  @
202 @@@@@@@@@@@@@@@@
203
204 n←800
205
206 FOR i ← 1 TO n
207     nodes[ i]  ← NEW node
208     node_number OF nodes[ i] ←i
209     JOIN nodes[ i]  TO node_list
210 END
211
212 supernode ← 1
213 lator_gast:
214 i ← READ FROM hconfig
215 supernode ← i
216
217 WHILE i > 0
218     THIS node ← nodes[ i]
219     @WAIT 1
220     nd_x      ← READ FROM hconfig
221     nd_y      ← READ FROM hconfig
222     IF  i≠1 & i≠20 & i≠41  & i≠60  & i≠81  & i≠100 & i≠501 & i≠520 & i≠541 & i≠56
223             & i≠20 & i≠600 & i≠201 & i≠218 & i≠231 & i≠248 & i≠261 & i≠278 & i≠29
224           & i≠338 & i≠351 & i≠368 & i≠381 & i≠398 & i≠411 & i≠428 & i≠441 & i≠45
225         THIS section ← NEW section
226         outs[ 1]  OF nodes[ i-1]  ← THIS section
227         end_point ← THIS node
228         s_length  ← distance_to(nd_x OF nodes[ i-1] ,→
229                                 nd_y OF nodes[ i-1] )
230         JOIN THIS section TO sections
231     END
232
233     IF (i ≥ 1 & i ≤ 118) | (i ≥ 501 & i ≤ 618)
234         nodespeed OF THIS node ← full_speed
235         exittime OF THIS node ← 2*(SQRT(((3/2)*2)/acc)) @3m is de breedte van een
236         nodedistance1 OF THIS node ← (full_speed*full_speed)/(2*acc)
237         nodedistance2 OF THIS node ← ((2*exittime OF THIS node)+(2*8)+2*(nodedist.
238
239     END
240     IF (i ≥ 201 & i ≤ 483)
241         nodespeed OF THIS node ← node_speed
242         nodedistance1 OF THIS node ← (node_speed*node_speed)/(2*acc)
243         nodedistance2 OF THIS node ← nodedistance1 OF THIS node
244         exittime OF THIS node ← 2*(SQRT(16/acc))                    @16 lengte va
245     END
246
247     n_nodes ← i
248     i ← READ FROM hconfig
```

In **line 263 to 273** the stacks are created

i is the number of stacks to be created

create_stack is a *macro*

In line 274 to 283 platforms are created

create_platform is a macro creating the platforms

In line 284 to 303 the short cuts are created (like the main tracks and the stacks and the platforms these consist of sections as well

In **line 304 to 321** supershort cuts are created these act the same as shortcuts but *are different because of their location at the terminal*

```
249 END
250
251 THIS node ← nodes[ n_nodes]
252 THIS section ← NEW section
253 outs[ 1]    ← THIS section
254 end_point ← nodes[ supernode]
255 s_length  ← distance_to(nd_x OF end_point,nd_y OF end_point)
256
257 JOIN THIS section TO sections
258
259 i ← READ FROM hconfig
260
261 GOTO lator_gast IF i>0
262
263 @@@@@@@@@@@@@@@@@@
264 @   stacks       @
265 @@@@@@@@@@@@@@@@@@
266
267 i←1
268 char ← READ FROM hconfig
269 WHILE char ≠ "END"
270     JUMP TO create_stack    @ supplier
271     char ← READ FROM hconfig
272 END
273
274 @@@@@@@@@@@@@@@@@@
275 @   platforms    @
276 @@@@@@@@@@@@@@@@@@
277
278 char ← READ FROM hconfig
279 WHILE char ≠ "END"
280     JUMP TO create_platform
281     char ← READ FROM hconfig
282 END
283
284 @@@@@@@@@@@@@@@@@@@@@@
285 @     short cuts     @
286 @@@@@@@@@@@@@@@@@@@@@@
287
288 i ← READ FROM hconfig
289 WHILE i > 0
290     THIS NODE     ← NODES[ I]
291     @WAIT 1
292     THIS section ← NEW section
293     sec_row       ← NEW SET WITH HISTORY
294     sec_row_number ←i
295     outs[ 2]      ← THIS section
296     side_track[ 1]    ← short_cut @  macro
297     i ← READ FROM hconfig
298     end_point ← nodes[ i]
299     s_length  ← distance_to(nd_x OF end_point,nd_y OF end_point)
300     JOIN THIS section TO sections
301     i ← READ FROM hconfig
302 END
303
304 @@@@@@@@@@@@@@@@@@@@@@@@@@
305 @    super short cuts     @
306 @@@@@@@@@@@@@@@@@@@@@@@@@@
307
308 i ← READ FROM hconfig
309 WHILE i > 0
310     THIS NODE     ← NODES[ I]
```

create_portout is a macro that creates all the portainer outportations

conversation is a macro that creates the control box

creating_sections is a macro that creates sections and attaches these sections to windows

Terminate cancels all the operations at the terminal

```
311     @WAIT 1
312     THIS section ← NEW section
313     sec_row      ← NEW SET
314     outs[ 3]     ← THIS section
315     side_track[ 2]    ← super_shortcut
316     i ← READ FROM hconfig
317     end_point ← nodes[ i]
318     s_length  ← distance_to(nd_x OF end_point,nd_y OF end_point)
319     JOIN THIS section TO sections
320     i ← READ FROM hconfig
321 END
322
323
324
325
326
327 @-------------------------------------------------@
328 @Start simulation by activating the ship generator@
329 @-------------------------------------------------@
330
331
332 anchoragetimes←NEW WINDOW CALLED "anchorage"
333 ATTACH HISTORY OF anchorage TO anchoragetimes
334
335 berthoccupied[ 1] ←NEW WINDOW CALLED "berth1occu"
336 ATTACH HISTORY OF berth[ 1] TO berthoccupied[ 1]
337
338 berthoccupied[ 2] ←NEW WINDOW CALLED "berth2occu"
339 ATTACH HISTORY OF berth[ 2] TO berthoccupied[ 2]
340
341 berthoccupied[ 3] ←NEW WINDOW CALLED "berth3occu"
342 ATTACH HISTORY OF berth[ 3] TO berthoccupied[ 3]
343
344 JUMP TO create_portout
345
346 JUMP TO conversation
347
348 JUMP TO creating_sections
349
350
351 portainersactive ← NEW WINDOW CALLED "active portainers"
352 ATTACH HISTORY OF unloadingship TO portainersactive
353
354 shipdwelltimes ← NEW WINDOW CALLED "shipdwelltimes"
355 ATTACH HISTORY OF loadingship TO shipdwelltimes
356
357 ACTIVATE shipgen FROM generate IN genmod
358
359 WAIT 30 HOURS
360
361 @-------------------------------------------------@
362 @End the simulation by cancelling the ship generator@
363 @-------------------------------------------------@
364
365 CANCEL shipgen
366
367
368 TERMINATE
369
```

The generator is responsible for generating the interarriveltimes of the vessels

Timebig refers to the interarrivaltime of a mega vessel
Timesmall refers to the interarivaltime of a feeder

if **line 7** is true this means there will be a feeder generated first
The generator waits the interarrival time before generating the feeder

Big is an attribute of ship of the type logical, "false" means this is a feeder
Anchorage is a set containing the waiting vessels

timebig < timesmall means the mega vessel will be generated first
the generator waits the interarrivaltime before generating a mega vessel

```
 1 generate:
 2
 3 timebig ←SAMPLE FROM arrivaltimesbig        @9000 @36*60        @SAMPLE FROM arri
 4 timesmall ← SAMPLE FROM arrivaltimes
 5
 6 begin:
 7 IF timebig ≥ timesmall
 8     WAIT timesmall
 9     timebig←timebig-timesmall
10     newship←NEW ship
11     shipnumber OF newship ← s
12
13     r←r+1
14     big OF newship ←FALSE @TRUE
15     JOIN newship TO anchorage
16
17     ACTIVATE newship FROM begin IN shipmod
18     s ← s+1
19     timesmall ← SAMPLE FROM arrivaltimes
20     REPEAT FROM begin
21 END
22
23 IF timebig < timesmall
24     WAIT timebig
25     timesmall←timesmall-timebig
26     newship←NEW ship
27     shipnumber OF newship ← s
28     big OF newship ← TRUE
29     JOIN newship TO anchorage
30     ACTIVATE newship FROM begin IN shipmod
31     s ← s+1
32     timebig ← SAMPLE FROM arrivaltimes@9000     @SAMPLE FROM arrivaltimes
33     REPEAT FROM begin
34 END
```

If the anchorage is not empty implies there is a vessel waiting with a berth request

Occupied[1] is a logical if occupied [1] is true berth1 is occupied by a vessel

If big is false this is a feeder and has to go to berth 1 or 3
The vessel can only go to berth 1 if this berth is not taken
In **line 9 to 20** the berthmaster assigns the free berth 1 to the feeder
the berthmaster tags berth 1 as taken

shipanchoragetime is an attribute of the vessel referring to the time in berth

If the vessel in the anchorage is a mega vessel berth 2 will be assigned

If the vessel in the anchorage is a feeder and berth 1 is already taken berth 3 will be assigned to the feeder

```
 1 begin:
 2
 3 IF anchorage IS NOT EMPTY
 4     berthmastership ← FIRST ship IN anchorage
 5     GOTO quayfull IF (occupied[ 1] =TRUE)& (occupied[ 2] =TRUE)& (occupied[ 3] =TRUE)
 6
 7     IF big OF berthmastership = FALSE
 8
 9         IF occupied[ 1] =FALSE
10             occupied[ 1] ← TRUE
11             shipberthnumber OF berthmastership ← 1
12             JOIN berthmastership TO unloadingship
13             JOIN berthmastership TO berth[ 1]
14             shipanchoragetime OF berthmastership ← NOW
15             JOIN berthmastership TO activeshiplist
16             REMOVE FIRST ship IN anchorage FROM anchorage
17             ACTIVATE agvmaster FROM byeshuffle IN agvmastmod IF agvmaster IS NOT .
18             GOTO begin
19         END
20
21     END
22
23     IF big OF berthmastership = TRUE
24
25         IF occupied[ 2] =FALSE
26             occupied[ 2] ← TRUE
27             shipberthnumber OF berthmastership ← 2
28             JOIN berthmastership TO unloadingship
29             JOIN berthmastership TO berth[ 2]
30             shipanchoragetime OF berthmastership ← NOW
31             JOIN berthmastership TO activeshiplist
32             REMOVE FIRST ship IN anchorage FROM anchorage
33             ACTIVATE agvmaster FROM byeshuffle IN agvmastmod IF agvmaster IS NOT .
34             GOTO begin
35         END
36
37     END
38     IF BiG OF berthmastership = FALSE @ TRUE
39     beter:
40         IF occupied[ 3] =FALSE
41             occupied[ 3] ← TRUE
42             shipberthnumber OF berthmastership   ← 3
43             JOIN berthmastership TO unloadingship
44             JOIN berthmastership TO berth[ 3]
45             shipanchoragetime OF berthmastership ← NOW
46             JOIN berthmastership TO activeshiplist
47             REMOVE FIRST ship IN anchorage FROM anchorage
48             ACTIVATE agvmaster FROM byeshuffle IN agvmastmod IF agvmaster IS NOT .
49             GOTO begin
50         END
51
52     END
53     quayfull:
54 END
55
56 ACTIVATE berthmastership FROM berthing IN shipmod IF berthmastership IS NOT ACTIV
57
58 PASSIVATE
59
```

Importgenerator generates the import containers

Exportgenerator generates the export containers

Goneshiplist is a set containing the vessels that have occupied the berths.

```
 1 begin:
 2 shiparrivaltime ← ARRIVALTIME OF THIS ship
 3
 4 firstshuffleagv OF THIS ship ← 1
 5
 6
 7 import:
 8 ACTIVATE importgenerator FROM begin IN importgen
 9 WAIT
10
11 export:
12 ACTIVATE exportgenerator FROM begin IN exportgen
13 WAIT
14
15 berthpermission:
16 ACTIVATE berthmaster FROM begin IN berthmastmod
17 WAIT
18
19 berthing:
20 @WAIT 1 HOUR @berthing
21 ACTIVATE portainermaster FROM begin IN portainermastmod
22 WAIT
23
24 leaving:
25 occupied[ shipberthnumber] ← FALSE
26 WAIT 1 SECONDS
27
28 shipdwell ← NOW - ARRIVALTIME OF THIS ship
29 shipturnaroundtime ← NOW - shipanchoragetime OF THIS ship
30
31 ENTER goneshiplist
32
33
34 REMOVE THIS ship FROM activeshiplist
35 REMOVE THIS ship FROM loadingship
36 REMOVE FIRST ship IN berth[ shipberthnumber] FROM berth[ shipberthnumber]
37 ACTIVATE berthmaster FROM begin IN berthmastmod
38 TERMINATE
39
```

The moment the portainermaster is activated he cancels all the portainers **line 3 to 8**

portainermasterportainer is an attribute of the portainermaster and a reference to portainer

The portainermaster removes all the portainers from the sets portainerperberth and portaineractivelist

In **line 29 to 252** The portainermaster allocates the portainers to their positions which depend on the occupancy of the berths. All the different options are treated In this scenario 10 portainers are used for 14 positions in the reference terminal 14 portainers are used

futlocnr is the future location number of the portainer. The new location of the portainer.

```
 1 begin:
 2 FOR a ← 1 TO 18 @10
 3     portainermasterportainer ← FIRST portainer IN quayrail WITH portainernumber =
 4     IF portainermasterportainer IS ACTIVE
 5         CANCEL portainermasterportainer
 6     END
 7 END
 8 FOR a ← 1 TO 18
 9     pmportloc ← FIRST portloc IN portlocations
10     REMOVE EACH portainer IN locationset OF pmportloc FROM locationset OF pmportl
11     REMOVE pmportloc FROM portlocations
12     JOIN pmportloc TO portlocations
13     REMOVE EACH portainer IN locationset OF pmportloc FROM locationset OF pmportl
14 END
15 REMOVE EACH portainer IN  portaineractivelist FROM portaineractivelist
16 FOR e ← 1 TO 3
17     REMOVE EACH portainer IN portainerperberth[ e]  FROM portainerperberth[ e]
18 END
19 IF (occupied[ 1] =FALSE)& (occupied[ 2] =FALSE) & (occupied[ 3] =FALSE)
20 WAIT
21 END
22 IF (occupied[ 1] =TRUE)& (occupied[ 2] =FALSE) & (occupied[ 3] =FALSE)
23     IF big OF FIRST OF berth[ 1] =FALSE
24         FOR a ← 3 TO 6
25             portainermasterportainer ← FIRST portainer IN quayrail WITH→
26              portainernumber =a
27             ptberthnr OF portainermasterportainer ← 1
28             ptmship ← FIRST OF berth[ ptberthnr OF portainermasterportainer]
29             futportainershipnb OF portainermasterportainer ← shipnumber→
30              OF ptmship
31             futlocnr OF portainermasterportainer ← a
32             JOIN portainermasterportainer TO portainerperberth[ 1] IF  totalload→
33             OF ptmship > 0
34             JOIN portainermasterportainer TO portaineractivelist IF  totalload→
35             OF ptmship > 0
36             ACTIVATE portainermasterportainer FROM begin IN portainermod →
37             IF portainermasterportainer IS NOT ACTIVE
38         END
39         FOR a ← 7 TO 12
40             portainermasterportainer ← FIRST portainer IN quayrail WITH→
41              portainernumber =a
42             ptberthnr OF portainermasterportainer ← 2
43             futlocnr OF portainermasterportainer ← a
44             ACTIVATE portainermasterportainer FROM begin IN portainermod→
45             IF portainermasterportainer IS NOT ACTIVE
46         END
47         FOR a ← 13 TO 16
48             portainermasterportainer ← FIRST portainer IN quayrail →
49             WITH portainernumber =a
50             ptberthnr OF portainermasterportainer ← 3
51             futlocnr OF portainermasterportainer ← a
52             ACTIVATE portainermasterportainer FROM begin IN portainermod→
53             IF portainermasterportainer IS NOT ACTIVE
54         END
55         activeport ← 4
56         ACTIVATE agvmaster FROM begin IN agvmastmod
57     END
58 END
59 IF (occupied[ 1] =FALSE)& (occupied[ 2] =TRUE) & (occupied[ 3] =FALSE)
60     IF big OF FIRST OF berth[ 2] =TRUE
61     FOR a ← 3 TO 6
62         portainermasterportainer ← FIRST portainer IN quayrail WITH →
```

```
63          portainernumber =a
64          ptberthnr OF portainermasterportainer ← 1
65          futlocnr OF portainermasterportainer ← a
66          ACTIVATE portainermasterportainer FROM begin IN portainermod →
67          IF portainermasterportainer IS NOT ACTIVE
68      END
69      FOR a ← 7 TO 12
70          portainermasterportainer ← FIRST portainer IN quayrail WITH→
71          portainernumber =a
72          ptberthnr OF portainermasterportainer ← 2
73          ptmship ← FIRST OF berth[ ptberthnr OF portainermasterportainer]
74          futportainershipnb OF portainermasterportainer ← shipnumber OF →
75          ptmship
76          futlocnr OF portainermasterportainer ← a
77          JOIN portainermasterportainer TO portainerperberth[ 2] IF  →
78          totalload  OF ptmship > 0
79          JOIN portainermasterportainer TO portaineractivelist IF→
80          totalload  OF ptmship > 0
81          ACTIVATE portainermasterportainer FROM begin IN portainermod→
82          IF portainermasterportainer IS NOT ACTIVE
83      END
84      FOR a ← 13 TO 16
85          portainermasterportainer ← FIRST portainer IN quayrail WITH→
86          portainernumber =a
87          ptberthnr OF portainermasterportainer ← 3
88          futlocnr OF portainermasterportainer ← a
89          ACTIVATE portainermasterportainer FROM begin IN portainermod→
90          IF portainermasterportainer IS NOT ACTIVE
91      END
92      activeport ← 6
93      ACTIVATE agvmaster FROM begin IN agvmastmod
94      END
95  END
96   IF (occupied[ 1] =FALSE) & (occupied[ 2] =FALSE) & (occupied[ 3] =TRUE)
97      IF big OF FIRST OF berth[ 3] =FALSE
98          FOR a ← 3 TO 6
99              portainermasterportainer ← FIRST portainer IN quayrail→
100              WITH portainernumber =a
101             ptberthnr OF portainermasterportainer ← 1
102             futlocnr OF portainermasterportainer ← a
103             ACTIVATE portainermasterportainer FROM begin IN→
104             portainermod IF portainermasterportainer IS NOT ACTIVE
105         END
106         FOR a ← 7 TO 12
107             portainermasterportainer ← FIRST portainer IN quayrail→
108             WITH portainernumber =a
109             ptberthnr OF portainermasterportainer ← 2
110             futlocnr OF portainermasterportainer ← a
111             ACTIVATE portainermasterportainer FROM begin IN portainermod→
112             IF portainermasterportainer IS NOT ACTIVE
113         END
114         FOR a ← 13 TO 16
115             portainermasterportainer ← FIRST portainer IN quayrail→
116             WITH portainernumber =a
117             ptberthnr OF portainermasterportainer ← 3
118             ptmship ← FIRST OF berth[ ptberthnr OF portainermasterportainer]
119             futportainershipnb OF portainermasterportainer ← shipnumber OF ptmshi
120             futlocnr OF portainermasterportainer ← a
121             JOIN portainermasterportainer TO portainerperberth[ 3] IF→
122             totalload  OF ptmship > 0
123             JOIN portainermasterportainer TO portaineractivelist IF→
124             totalload  OF ptmship > 0
```

```
125             ACTIVATE portainermasterportainer FROM begin IN portainermod→
126             IF portainermasterportainer IS NOT ACTIVE
127         END
128         activeport ← 4
129         ACTIVATE agvmaster FROM begin IN agvmastmod
130     END
131 END
132 IF (occupied[ 1] = TRUE)& (occupied[ 2] =TRUE) & (occupied[ 3] =FALSE)
133     IF big OF FIRST OF berth[ 1] =FALSE & big OF FIRST OF berth[ 2] =TRUE
134     FOR a ← 3 TO 6
135         portainermasterportainer ← FIRST portainer IN quayrail→
136         WITH portainernumber = a
137         ptberthnr OF portainermasterportainer ← 1
138         futlocnr OF portainermasterportainer ← a
139         ptmship ← FIRST OF berth[ ptberthnr OF portainermasterportainer]
140         futportainershipnb OF portainermasterportainer ← shipnumber OF ptmship
141         JOIN portainermasterportainer TO portainerperberth[ 1] IF →
142         totalload  OF ptmship > 0
143         JOIN portainermasterportainer TO portaineractivelist IF →
144         totalload  OF ptmship > 0
145         ACTIVATE portainermasterportainer FROM begin IN portainermod→
146         IF portainermasterportainer IS NOT ACTIVE
147     END
148     FOR a ← 7 TO 12
149         portainermasterportainer ← FIRST portainer IN quayrail WITH →
150         portainernumber =a
151         ptberthnr OF portainermasterportainer ← 2
152         ptmship ← FIRST OF berth[ ptberthnr OF portainermasterportainer]
153         futportainershipnb OF portainermasterportainer ← shipnumber OF ptmship
154         futlocnr OF portainermasterportainer ← a
155         JOIN portainermasterportainer TO portainerperberth[ 2] IF →
156          totalload  OF ptmship > 0
157         JOIN portainermasterportainer TO portaineractivelist IF→
158           totalload  OF ptmship > 0
159         ACTIVATE portainermasterportainer FROM begin IN portainermod→
160          IF portainermasterportainer IS NOT ACTIVE
161     END
162     FOR a ← 13 TO 16
163         portainermasterportainer ← FIRST portainer IN quayrail →
164         WITH portainernumber =a
165         ptberthnr OF portainermasterportainer ← 3
166         futlocnr OF portainermasterportainer ← a
167         ACTIVATE portainermasterportainer FROM begin IN portainermod →
168         IF portainermasterportainer IS NOT ACTIVE
169     END
170     activeport ← 10
171     ACTIVATE agvmaster FROM begin IN agvmastmod
172     END
173 END
174 IF (occupied[ 1] =FALSE)& (occupied[ 2] =TRUE) & (occupied[ 3] =TRUE)
175     IF big OF FIRST OF berth[ 2] =TRUE & big OF FIRST OF berth[ 3] =FALSE
176         FOR a ← 3 TO 6
177             portainermasterportainer ← FIRST portainer IN quayrail→
178              WITH portainernumber =a
179             ptberthnr OF portainermasterportainer ← 1
180             futlocnr OF portainermasterportainer ← a
181             ACTIVATE portainermasterportainer FROM begin IN portainermod→
182              IF portainermasterportainer IS NOT ACTIVE
183         END
184         FOR a ← 7 TO 12
185             portainermasterportainer ← FIRST portainer IN quayrail WITH→
186              portainernumber =a
```

```
187             ptberthnr OF portainermasterportainer ← 2
188             ptmship ← FIRST OF berth[ ptberthnr OF portainermasterportainer]
189             futportainershipnb OF portainermasterportainer ← shipnumber OF ptmshi
190             futlocnr OF portainermasterportainer ← a
191             JOIN portainermasterportainer TO portainerperberth[ 2] IF →
192              totalload  OF ptmship > 0
193             JOIN portainermasterportainer TO portaineractivelist IF →
194              totalload  OF ptmship > 0
195             ACTIVATE portainermasterportainer FROM begin IN portainermod→
196              IF portainermasterportainer IS NOT ACTIVE
197         END
198         FOR a ← 13 TO 16
199             portainermasterportainer ← FIRST portainer IN quayrail WITH→
200              portainernumber =a
201             ptberthnr OF portainermasterportainer ← 3
202             ptmship ← FIRST OF berth[ ptberthnr OF portainermasterportainer]
203             futportainershipnb OF portainermasterportainer ← shipnumber OF ptmshi
204             futlocnr OF portainermasterportainer ← a
205             JOIN portainermasterportainer TO portainerperberth[ 3] IF→
206               totalload  OF ptmship > 0
207             JOIN portainermasterportainer TO portaineractivelist IF→
208               totalload  OF ptmship > 0
209             ACTIVATE portainermasterportainer FROM begin IN portainermod →
210             IF portainermasterportainer IS NOT ACTIVE
211         END
212         activeport ← 10
213         ACTIVATE agvmaster FROM begin IN agvmastmod
214     END
215 END
216 IF (occupied[ 1] =TRUE) & (occupied[ 2] =FALSE) & (occupied[ 3] =TRUE)
217     IF big OF FIRST OF berth[ 1] = FALSE & big OF FIRST OF berth[ 3] =FALSE
218         FOR a ← 3 TO 6
219             portainermasterportainer ← FIRST portainer IN quayrail→
220              WITH portainernumber = a
221             ptberthnr OF portainermasterportainer ← 1
222             ptmship ← FIRST OF berth[ ptberthnr OF portainermasterportainer]
223             futportainershipnb OF portainermasterportainer ← shipnumber OF ptmshi
224             futlocnr OF portainermasterportainer ← a
225             JOIN portainermasterportainer TO portainerperberth[ 1] IF→
226               totalload  OF ptmship > 0
227             JOIN portainermasterportainer TO portaineractivelist IF →
228              totalload  OF ptmship > 0
229             ACTIVATE portainermasterportainer FROM begin IN portainermod→
230              IF portainermasterportainer IS NOT ACTIVE
231         END
232         FOR a ← 7 TO 12
233             portainermasterportainer ← FIRST portainer IN quayrail→
234              WITH portainernumber = a
235             ptberthnr OF portainermasterportainer ← 2
236             futlocnr OF portainermasterportainer ← a
237             ACTIVATE portainermasterportainer FROM begin IN →
238             portainermod IF portainermasterportainer IS NOT ACTIVE
239         END
240         FOR a ← 13 TO 16
241             portainermasterportainer ← FIRST portainer IN quayrail →
242             WITH portainernumber = a
243             ptberthnr OF portainermasterportainer ← 3
244             ptmship ← FIRST OF berth[ ptberthnr OF portainermasterportainer]
245             futportainershipnb OF portainermasterportainer ← shipnumber OF ptmshi
246             futlocnr OF portainermasterportainer ← a
247             JOIN portainermasterportainer TO portainerperberth[ 3] IF→
248               totalload  OF ptmship > 0
```

```
249             JOIN portainermasterportainer TO portaineractivelist IF→
250               totalload  OF ptmship > 0
251             ACTIVATE portainermasterportainer FROM begin IN portainermod→
252              IF portainermasterportainer IS NOT ACTIVE
253         END
254         activeport ← 8
255         ACTIVATE agvmaster FROM begin IN agvmastmod
256     END
257 END
258 IF (occupied[ 1] =TRUE)& (occupied[ 2] =TRUE) & (occupied[ 3] =TRUE)
259     IF big OF FIRST OF berth[ 1] = FALSE & big OF FIRST OF berth[ 2] =TRUE→
260      & big OF FIRST OF berth[ 3] =FALSE
261     FOR a ← 3 TO 6
262         portainermasterportainer ← FIRST portainer IN quayrail→
263          WITH portainernumber = a
264         ptberthnr OF portainermasterportainer ← 1
265         ptmship ← FIRST OF berth[ ptberthnr OF portainermasterportainer]
266         futportainershipnb OF portainermasterportainer ← shipnumber OF ptmship
267         futlocnr OF portainermasterportainer ← a
268         JOIN portainermasterportainer TO portainerperberth[ 1] IF→
269           totalload  OF ptmship > 0
270         JOIN portainermasterportainer TO portaineractivelist→
271              IF totalload  OF ptmship > 0
272         ACTIVATE portainermasterportainer FROM begin IN portainermod→
273          IF portainermasterportainer IS NOT ACTIVE
274     END
275     FOR a ← 7 TO 12
276         portainermasterportainer ← FIRST portainer IN quayrail→
277          WITH portainernumber = a
278         ptberthnr OF portainermasterportainer ← 2
279         ptmship ← FIRST OF berth[ ptberthnr OF portainermasterportainer]
280         futportainershipnb OF portainermasterportainer ← shipnumber OF ptmship
281         futlocnr OF portainermasterportainer ← a
282         JOIN portainermasterportainer TO portainerperberth[ 2] IF →
283          totalload  OF ptmship > 0
284         JOIN portainermasterportainer TO portaineractivelist IF→
285           totalload  OF ptmship > 0
286         ACTIVATE portainermasterportainer FROM begin IN portainermod→
287          IF portainermasterportainer IS NOT ACTIVE
288     END
289     FOR a ← 13 TO 16
290         portainermasterportainer ← FIRST portainer IN quayrail→
291          WITH portainernumber = a
292         ptberthnr OF portainermasterportainer ← 3
293         ptmship ← FIRST OF berth[ ptberthnr OF portainermasterportainer]
294         futportainershipnb OF portainermasterportainer ← shipnumber OF ptmship
295         futlocnr OF portainermasterportainer ← a
296         JOIN portainermasterportainer TO portainerperberth[ 3] IF→
297          totalload OF ptmship > 0
298         JOIN portainermasterportainer TO portaineractivelist IF →
299         totalload OF ptmship > 0
300         ACTIVATE portainermasterportainer FROM begin IN portainermod→
301          IF portainermasterportainer IS NOT ACTIVE
302     END
303     activeport ← 14
304     ACTIVATE agvmaster FROM begin IN agvmastmod
305     END
306 END
307 PASSIVATE
```

pal is portainer active list

This means there are portainers active

if actlocnr is not futlocnr the portainer is not yet at the right location. From the moment the portainer is at its right location it can get rgv's assigned again

In **line 20 to 40** the agvmaster looks what the portainers are doing and determines the number of needed rgv's for the portainers

From **line 46 to 77** the right number of rgv's per portainer are activated

Bye is an attribute of the rgv of the type logical. If bye of a rgv is true it has to go to the parking the moment it has finished the assignment it is working on

In line **54 to 77** extra rgv's are created

```
 1 begin:
 2
 3 @determine the number of active portainers
 4
 5 portaineropplek:
 6 pal ← LENGTH OF portaineractivelist
 7 IF LENGTH OF portaineractivelist = 0
 8     GOTO beter2
 9 END
10
11 WHILE pal ≠ 0
12 agvportainer←FIRST portainer IN portaineractivelist
13 REMOVE agvportainer FROM portaineractivelist
14 JOIN agvportainer TO portaineractivelist
15
16     IF actlocnr OF agvportainer ≠ futlocnr OF agvportainer
17         agvppneeded←0
18         GOTO beter
19     END
20     IF ptship OF agvportainer BELONGS TO unloadingship & actlocnr OF→
21     agvportainer = futlocnr OF agvportainer
22     agvppneeded←6  @ even tijdelijk aagepast
23     pp ← agvppneeded - (LENGTH OF agvppactive OF agvportainer +→
24     LENGTH OF agvppactiver OF agvportainer)
25     END
26     IF  ptship OF agvportainer BELONGS TO loadingship & actlocnr OF
27         agvportainer = futlocnr OF agvportainer& firstshuffleagv OF
28         ptship OF agvportainer = 1
29     agvppneeded←12  @ even tijdelijk aagepast
30     pp←agvppneeded - (LENGTH OF agvppactive OF agvportainer +→
31     LENGTH OF agvppactiver OF agvportainer)
32     END
33     IF  ptship OF agvportainer BELONGS TO loadingship & actlocnr OF→
34         agvportainer = futlocnr OF agvportainer& firstshuffleagv OF
35         ptship OF agvportainer ≠ 1
36         @GOTO beter
37         agvppneeded←8  @ even tijdelijk aagepast
38         pp←agvppneeded - (LENGTH OF agvppactive OF agvportainer +→
39          LENGTH OF agvppactiver OF agvportainer)
40     END
41
42     IF pp = 0 @er zijn genoeg agv's
43     GOTO beter
44     END
45
46     IF pp<0 @er zijn teveel agv's
47         FOR i←1 TO (-1*pp)
48         byeagv ← FIRST agv IN agvppactive OF agvportainer
49         REMOVE byeagv FROM agvppactive OF agvportainer
50         JOIN byeagv TO agvppactive OF agvportainer
51         bye OF byeagv ← TRUE
52         END
53     END
54     IF pp>0          @er zijn te weinig agv's
55         FOR i←1 TO pp
56             IF LENGTH OF agvppactive OF agvportainer=0
57             hiagv←FIRST agv IN agvparking WITH (agvnumber)MODULO(2)=1
58             JOIN hiagv TO agvppactive_odd OF agvportainer
59             REMOVE hiagv FROM agvparking
60             GOTO vreter
61         END
62
```

In **line 66 to 76** it is made sure there are just as much odd as even rgv's
If ri/ra >0.5 this means there are more odd rgv's so an even rgv must be activated

if ri/ra <0.5 this means there are more even rgv's so an odd rgv must be activated

In line **78** to **100** the start conditions for the rgv's are created

The starting position of the rgv is chosen

From **line 105** to the end is a separate part of the agv master which has nothing to do with the precedent

```
63          WHILE LENGTH OF agvppactive_odd OF agvportainer≠0
64              ri←LENGTH OF agvppactive_odd OF agvportainer
65              ra←LENGTH OF agvppactive OF agvportainer
66              IF (ri/ra)>0.5
67                  HIAGV←FIRST agv IN agvparking WITH (agvnumber)MODULO(2)=0
68                  REMOVE hiagv FROM agvparking
69                  GOTO vreter
70              END
71              IF (ri/ra)≤0.5
72                  hiagv← FIRST agv IN agvparking WITH (agvnumber)MODULO(2)=1
73                  JOIN hiagv TO agvppactive_odd OF agvportainer
74                  REMOVE hiagv FROM agvparking
75                  GOTO vreter
76              END
77      END
78      vreter:
79          @REMOVE FIRST agv IN agvparking FROM agvparking
80          JOIN hiagv TO activeagvlist
81          JOIN hiagv TO agvworklist
82          JOIN hiagv TO agvppactive OF agvportainer
83          agvberthnb OF hiagv ← shipberthnumber OF ptship OF agvportainer
84          yard OF hiagv←TRUE
85          ro←LENGTH OF agvppactive OF agvportainer
86          ru←CEIL(ro/2)
87          ro←(ru)MODULO(2)
88              IF ro=0
89                  cont20shuf OF hiagv ←TRUE
90              END
91              IF ro =1
92                  cont20shuf OF hiagv ← FALSE
93              END
94          x OF hiagv ← futlocnr OF agvportainer
95          y OF hiagv← CEIL((x OF hiagv)/2)*20 - 18 +→
96           ((-1 + (x OF hiagv))MODULO(2))*9
97          yy OF hiagv ← y OF hiagv
98          ACTIVATE hiagv FROM begin IN agvmod
99          END
100     END
101 beter:
102 pal←pal-1
103 END
104
105 beter2:
106 WAIT
107
108 byeshuffle:
109 amship ← LAST ship IN activeshiplist
110
111 IF LENGTH OF agvberthshufflelist[ shipberthnumber OF amship] >0
112     FOR i←1 TO LENGTH OF agvberthshufflelist[ shipberthnumber OF amship]
113         byeagv ← FIRST agv IN agvberthshufflelist[ shipberthnumber OF amship]
114         bye OF byeagv←TRUE
115     END
116 END
117
118 PASSIVATE
```

Line **3 to 70** is about the positioning of the portainer at the quay

Ptportloc is the portainerlocation the portainer occupies at the moment

if the length of assignedrgv's of the portainerlocation >0 means there is a rgv on its way to be served by this particular portainer at that particular place. So it is for the portainer impossible to move to another location it will first have to serve this rgv

In contrast to **line 11** there are no rgv's assigned. The portainer looks weather it is at the right location (inpositioncheck).
If in **line 23** the condition counts the portainer is at the right location

If the condition in **line 41** counts the portainer has to move to a portlocation with a higher number. The portainer waits until the adjacent place is free

If the condition in **line 56** counts the portainer has to move to a portlocation with a lower number

```
 1 begin:
 2
 3 @@@@
```

From **line 72** the actual loading and unloading of the rgv's starts

Portlocqueue is a set containing rgv's waiting under the portainer to be served
Portasqueue is a set containing rgv's assigned to that portlocation. These rgv' s could be empty wanting a container from the portainer

Portainertimes is a macro creating the portainercycletimes
qq is the portainercycletime calculated in portainertimes
Line 75 to 81 a rgv is assigned to the portainer this does not automatically means the rgv is already at the portainerlocation it could be on the way

Line 83 to 90 a rgv has arrived under the portainer in the queue

Line 92 to 109 deals with the unloading of the vessel

The portainer has already worked qq seconds to get the container from the vessel
the 10 seconds in **line 98** are for the actual positioning of the container on the rgv
The portainer removes the rgv from its queue and prepares the rgv to leave the portainerlocation

Line 110 to 135 deal with the loading of the vessels

The portainer removes the container from the rgv this takes 10 seconds hereafter the rgv is able to move on while the portainer has to do its work
In **line 120** the portainer keeps count so it knows when the vessel can leave the berth
Line 122 to 129 prepare the rgv to leave the portainerlocation

```
63          actlocnr OF THIS portainer ← (actlocnr OF THIS portainer-1)
64          ptportloc ← FIRST portloc IN portlocations WITH →
65          portlocnumber = actlocnr
66          ENTER locationset OF ptportloc IF THIS portainer→
67           BELONGS NOT TO locationset OF ptportloc
68          quaylocation[ actlocnr]   ← TRUE
69          GOTO inpositioncheck
70      END
71  END
72  restart:
73  WAIT WHILE portlocqueue OF ptportloc IS EMPTY & LENGTH→
74       OF portasqueue OF THIS portainer = 0
75  IF LENGTH OF portasqueue OF THIS portainer ≠ 0
76      asagv ← FIRST agv IN portasqueue OF THIS portainer
77      JUMP TO portainertimes
78      WAIT (QQ OF THIS portainer) SECONDS
79      REMOVE asagv FROM portasqueue OF THIS portainer
80      WAIT WHILE portlocqueue OF ptportloc IS EMPTY
81  END
82
83  portagv ← FIRST agv IN portlocqueue OF ptportloc→
84   WITH x = actlocnr OF THIS portainer
85  IF loadedagv OF portagv = TRUE
86      GOTO load @ is het laden van het schip
87  END
88  IF loadedagv OF portagv = FALSE
89      GOTO unload
90  END
91
92  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
93  @the unloading of the vessel@
94  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
95
96  unload:
97  IF  portlocqueue OF ptportloc IS NOT EMPTY
98          WORK 10 SECONDS
99          loadedagv OF portagv ← TRUE
100         REMOVE portagv FROM portlocqueue OF ptportloc
101         REMOVE portagv FROM assignedagvs OF ptportloc→
102          IF portagv BELONGS TO assignedagvs OF ptportloc
103         ACTIVATE portagv FROM begin IN agvmod
104         JOIN portagv TO agvworklist @ deze zit er nieuw in
105 END
106 IF assignedagvs OF ptportloc IS EMPTY
107     GOTO inpositioncheck
108 END
109 GOTO restart
110
111 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
112 @the loading of the vessels@
113 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
114
115 load:
116
117 IF loadedagv OF portagv = TRUE
118     IF portlocqueue OF ptportloc IS NOT EMPTY
119         WORK 10 SECONDS
120         totalload OF ptship ← (totalload OF ptship) -1
121         loadedagv OF portagv ← FALSE
122         REMOVE portagv FROM portlocqueue OF ptportloc
123         REMOVE portagv FROM assignedagvs OF ptportloc IF→
124          portagv BELONGS TO assignedagvs OF ptportloc
```

This is where the portainer works the container to its place on/in the vessel

If totalload = 0, This means the vessel is completely loaded and ready to leave
The portainer that has delivered the last container tells the vessel to leave

```
125         JOIN portagv TO agvworklist
126         yard OF portagv ←FALSE
127         ACTIVATE portagv FROM begin IN agvmod
128         asagv ← FIRST agv IN assignedagvs OF ptportloc
129             IF totalload OF ptship = 0
130                 GOTO ietstevroeg
131             END
132         JUMP TO portainertimes
133         WORK (qq OF THIS portainer) SECONDS
134     END
135 END
136 ietstevroeg:
137 IF  totalload OF ptship = 0
138     ACTIVATE ptship FROM leaving IN shipmod IF ptship IS NOT ACTIVE
139 END
140 IF assignedagvs OF ptportloc IS EMPTY
141     GOTO inpositioncheck
142 END
143 GOTO begin
144
145
146
```

the exportgen creates the exportcontainers for the vessels

If big is true this vessel is a mega vessel

If big is false this vessel is a feeder

totalload is the number of containers the vessel has to receive at the terminal here this number is deterministic. Totalload starts at the maximum when it is 0 the vessel can leave the terminal
totalu is the start value of totalload, this value doesn't change in contrast to totalload

propex40 is the share of 40 foot containers

```
 1 begin:
 2
 3 exportship ← THIS ship  @FIRST ship IN anchorage
 4 IF big OF exportship = TRUE
 5 ww←1575
 6 END
 7 IF big OF exportship = FALSE
 8 ww←525
 9 END
10
11 totalload OF exportship ← ww
12 totalu OF exportship ← ww
13 ex40 OF exportship ← 0.5*ww
14 ex20 OF exportship ← 0.5*ww
15
16
17 propex40 ← ex40 OF exportship/totalload OF exportship
18 propex20 ← ex20 OF exportship/totalload OF exportship
19
20
21
22
23 ACTIVATE exportship FROM berthpermission IN shipmod IF exportship IS NOT ACTIVE
24
25 WAIT
```

The importgen creates the import containers for the vessels

```
 1 begin:
 2
 3 importship ← THIS ship  @FIRST ship IN anchorage
 4
 5 IF big OF importship = TRUE
 6     ww←18
 7 END
 8 IF big OF IMPORTSHIP = FALSE
 9     ww←20
10 END
11
12
13 totalunload OF importship ← ww
14 total OF importship ← ww
15 im40 OF importship ← 0.5*ww
16 im20 OF importship ← 0.5*ww
17
18 propim40 OF importship ← (im40 OF importship)/(totalunload OF importship)
19 propim20 OF importship ← (im20 OF importship)/(totalunload OF importship)
20
21
22
23
24 ACTIVATE importship FROM export IN shipmod IF IMPORTSHIP IS NOT ACTIVE
25
26 WAIT
```

The first thing for the rgv is to activate the trafficmaster and ask for a assignment. The trafficmaster will looking at the attributes of the rgv activate the rgv from a specific point in the agv module

The trafficmaster activates the rgv from shuffling, toyard or uneasy etc. it all starts in **line 8 to 10**

preclaimnodes is a macro that looks whether it is possible for the rgv to enter the main track and start driving. This macro enables the rgv to enter the main track without interfering with other rgv's. The rgv only returns from this macro when the main track the rgv wants to enter is not claimed by another rgv
line 15 the rgv returns from the macro and claims the entrance node
exittime is a specific time for a rgv to pull up and leave a node

wheel rotation of the rgv takes 4 seconds

This is the driving on the node **line 25**

entrance node is not needed anymore and removed from the set entrance **line 25**
exit is the first node where the rgv has to leave the main track it has just entered

In **line 31 to 53** the actual driving of the rgv takes place this is a great loop
line 32 side_track is a macro, this can be a macro "short cut" or "stack_crane" or "super shortcut" in **line 32 to 38** there are six the same statements concerning the the side_track this is because it is possible a rgv has to take a few short cuts in a row

line 38 to 43 concerns the driving on the main track

line 42 is the driving of the current section minus the time needed for accelerating and slowing down these times are already deducted in **line 40 and 46**

```
 1 begin:
 2
 3 WAIT WHILE trafficmaster IS ACTIVE
 4 ACTIVATE trafficmaster FROM begin IN trafficmastmod IF trafficmaster→
 5  IS NOT ACTIVE
 6 WAIT
 7 @--------------------------------------------------------------------------
 8 shuffling:
 9 toyard:
10 uneasy:
11 toquay:
12 go_ahead:
13
14 entrance_node ← FIRST node IN entrance OF THIS agv
15 JUMP TO preclaimnodes
16
17 claimed OF entrance_node ← claimed OF entrance_node+1
18 cur_node←entrance_node
19 WORK exittime OF cur_node
20
21 LEAVE stack_row OF agvstack OF THIS agv
22
23 WORK 8 SECONDS
24
25 WORK (nodedistance1 OF cur_node)/(nodespeed OF cur_node)
26
27 REMOVE entrance_node FROM entrance OF THIS agv
28 exit ← FIRST node IN route OF THIS agv
29 REMOVE FIRST node IN route OF THIS agv  FROM route OF THIS agv
30
31 op_naar_het_stack:
32     JUMP TO side_track[ 1] OF cur_node IF cur_node IS exit
33     JUMP TO side_track[ 1] OF cur_node IF cur_node IS exit
34     JUMP TO side_track[ 1] OF cur_node IF cur_node IS exit
35     JUMP TO side_track[ 1] OF cur_node IF cur_node IS exit
36     JUMP TO side_track[ 1] OF cur_node IF cur_node IS exit
37     JUMP TO side_track[ 1] OF cur_node IF cur_node IS exit
38     JUMP TO side_track[ 2] OF cur_node IF cur_node IS superexit
39     cur_sec←outs[ 1] OF cur_node
40     WORK (nodedistance1 OF cur_node)/(nodespeed OF cur_node)
41     claimed OF cur_node←claimed OF cur_node-1
42     WORK MAX(0,(s_length OF cur_sec-(nodedistance1 OF→
43      cur_node+nodedistance2 OF cur_node)))/(nodespeed OF cur_node)
44     lastnode OF THIS agv ← cur_node
45     cur_node ←end_point OF cur_sec
46     claimed OF cur_node←claimed OF cur_node+1
47     WORK (nodedistance2 OF cur_node)/(nodespeed OF cur_node)
48     JUMP TO side_track[ 1] OF cur_node IF cur_node IS exit
49     JUMP TO side_track[ 1] OF cur_node IF cur_node IS exit
50     JUMP TO side_track[ 1] OF cur_node IF cur_node IS exit
51     JUMP TO side_track[ 1] OF cur_node IF cur_node IS exit
52     JUMP TO side_track[ 2] OF cur_node IF cur_node IS superexit
53 REPEAT FROM op_naar_het_stack
54 ga_maar_in_de_rij_staan:
55 ACTIVATE agvcrane FROM begin IN cranemod IF agvcrane IS NOT ACTIVE
56 WAIT
57
58 toparking:
59 JUMP TO agv_toparking
60 @--------------------------------------------------------------------------
61
```

Agvworklist a set containing all the rgv's requesting the trafficmaster for work

If the agv has decided the rgv has to go because there are to many rgv's active he will give an attribute bye, this is spotted by the trafficmaster and he will send the rgv to the parking **line 6 to 10**

If shuffleagv is true it means the rgv is shuffling and does not have to work for a portainer any more it has to be removed from ppactive

This is the case when a rgv is working for a portainer that has just finished a vessel and the rgv does not have to shuffle

The rgv has asked the trafficmaster for an assignment but the portainer this rgv is working for is not at the right location

In **line 38 to 76** a rgv calls from the yard and the vessel is unloading

If the conditions in **line 42** are correct the trafficmaster identifies that the rgv calls from the yard and the vessel it is working for is unloading

In line 48 to 54 the totalunload =0 the vessel is unloaded and has to be loaded

If inplacecounter >0 this would mean the vessel can be loaded with containers from the same berth. This is possible If some shuffling is done beforehand

agv_load_back is a macro determining the route the rgv has to take from the yard to the right portainerlocation

The rgv will go to the portainerlocation and is assigned, this enables the portainer to have a container ready when the rgv arrives.

```
 1 begin:
 2 tmagv ← FIRST agv IN agvworklist
 3 tmship ← FIRST ship IN activeshiplist WITH shipberthnumber=agvberthnb OF tmagv
 4 tmport← FIRST portainer IN quayrail WITH actlocnr = x OF tmagv
 5
 6     IF BYE OF TMAGV = TRUE & LOADEDAGV OF TMAGV =FALSE
 7         REMOVE TMAGV FROM AGVWORKLIST
 8         ACTIVATE TMAGV FROM TOPARKING IN AGVMOD
 9         WAIT
10     END
11     IF shuffleagv OF tmagv = TRUE
12         REMOVE tmagv FROM agvppactive OF tmport
13         JOIN tmagv TO agvppactiver OF tmport
14         GOTO shuffleprocedure
15     END
16
17     IF tmship IS NONE
18         ACTIVATE tmagv FROM toparking IN agvmod IF tmagv IS NOT ACTIVE
19         REMOVE tmagv FROM agvworklist
20         WAIT
21     END
22
23     IF tmport IS NONE
24         ACTIVATE tmagv FROM toparking IN agvmod IF tmagv IS NOT ACTIVE
25         REMOVE tmagv FROM agvworklist
26         loadedagv OF tmagv ← FALSE
27     WAIT
28     END
29
30     IF actlocnr OF tmport ≠ futlocnr OF tmport & loadedagv OF tmagv = FALSE
31         ACTIVATE tmagv FROM toparking IN agvmod  IF tmagv IS NOT ACTIVE
32         REMOVE tmagv FROM agvworklist
33         WAIT
34     END
35
36 restart:
37
38 @@@@
```

Line 74 to 114 deals with empty rgv's calling from the yard when there is a vessel loading

tmtotalex is the number of containers that still have to go with the vessel

If the condition in **line 83** counts all the containers are on board of the vessel and the shuffling at the terminal can start

In **line 88 to 96** the rgv is being prepared for the shuffling

Loadyardallocation is a macro that chooses where the rgv has to get a load container

agv_stack_stack is a macro that tells the rgv how to get to the right stack (route)

The rgv is being assigned to the portlocation because it has a assignment to eventually deliver a container to a portainer so it is important the portainer knows the rgv is coming

In **line 116 to 137** a loaded rgv calls from the yard and its vessel is loading

```
63      OF tmport
64      ACTIVATE tmport FROM restart IN portainermod IF tmport IS NOT ACTIVE
65      IF tmagv IS ACTIVE
66          CANCEL tmagv
67      END
68      ACTIVATE tmagv FROM toquay IN agvmod IF tmagv IS NOT ACTIVE
69      REMOVE tmagv FROM agvworklist
70      WAIT
71  END
72
73
74  @@@@
```

agv_stack_berth is a macro that creates the route from the stack to the right berth

Line 138 to 157 is about a empty rgv that calls from the quay with a vessel loading

unloadyardallocation is a macro that decides where the rgv has to deliver the container
agv_load creates the route for the rgv to get to the stack

In **line 171 to 184** if the condition counts it would mean the vessel is loaded and the rgv has got to be prepared for the shuffling

```
125      shipberthnumber=agvberthnb OF tmagv
126     JUMP TO agv_stack_berth
127
128     REMOVE tmagv FROM agvworklist
129     IF tmagv IS ACTIVE
130         CANCEL tmagv
131     END
132     IF TMAGV IS ACTIVE
133         CANCEL TMAGV
134     END
135     ACTIVATE tmagv FROM toquay IN agvmod IF tmagv IS NOT ACTIVE
136     WAIT
137 END
138 @@@@
```

Loadyardallocation is a macro that chooses where the rgv has to get a load container

agv_stack_berth is a macro that creates the route from the stack to the right berth

In **line 200 to 213** the right stackcrane is determined
This formula makes it possible to vary the number of cranes at the terminal

230 to 249 deals with a loaded rgv calling from the portainer and its vessel is loading
This looks like a strange situation, this rgv has just received the last unload container and now asks the trafficamster what to do. This is the transition between unloading and loading

agv_lossen creates the route for the rgv to get to the stack

```
187     JUMP TO loadyardallocation
188
189     IF (agvnumber OF tmagv)MODULO(2) = 0 & (uneasy OF tmagv = TRUE)
190         JUMP TO agv_uneasy_b_s
191         GOTO skip
192     END
193
194     JUMP TO agv_berth_stack
195
196     skip:
197
198     REMOVE tmship FROM loadingship
199     JOIN tmship TO loadingship
200 yyy OF tmagv←y OF tmagv
201 xxx OF tmagv←FLOOR((yyy OF tmagv-1)/20)
202 zzz OF tmagv←FLOOR((yyy OF tmagv-1)/20)
203 yyy OF tmagv←(y OF tmagv)-(zzz OF tmagv)*20
204
205 IF (yyy OF tmagv)MODULO(2)=0
206     yyy OF tmagv ← yyy OF tmagv -1
207 END
208 @IF (yyy OF tmagv)> 10
209     @yyy OF THIS agv← yyy OF tmagv -10
210 @END
211
212 yyy OF tmagv ← yyy OF tmagv + 20*(xxx OF tmagv)
213     tmcrane ← FIRST crane IN cranelist WITH cranenumber = yyy OF tmagv
214     JOIN tmagv TO assignedqueue OF tmcrane
215
216     REMOVE tmcrane FROM cranelist
217     JOIN tmcrane TO cranelist
218     REMOVE tmagv FROM agvworklist
219     agvportloc ← FIRST portloc IN portlocations WITH→
220          portlocnumber = actlocnr OF tmport
221     JOIN tmagv TO assignedagvs OF agvportloc IF →
222         tmagv BELONGS NOT TO assignedagvs OF agvportloc
223     IF TMAGV IS ACTIVE
224         CANCEL TMAGV
225     END
226     ACTIVATE tmagv FROM toyard IN agvmod IF tmagv IS NOT ACTIVE
227 WAIT
228 END
229
230 @@@@
```

From line 250 to the end has everything to do with the shuffling at the terminal

an empty rgv is under its portainer or crane and about to retrieve a container from the stack behind the portainer for the first time.

yardshuffleallocation is a macro that decides where to collect a shuffle container

If this condition counts there are no more containers to be shuffled and the rgv can go to the parking

If this condition counts there are shuffle containers and one is deducted

odd rgv's take a different route than even rgv's
agv_berth_stack creates the route from the berth to the stack

agv_uneasy_b_s creates a route for the rgv from berth to stack

y_creation is a macro responsible for determining the right stacking crane as in **line 200**

empty rgv under a crane and about to retrieve a container from the current stack behind the portainer from **line 306 to 340**

```
249
250 shuffleprocedure:
251
252 @@@@
```

```
311     @@@@
```

```
373     IF loadedagv OF tmagv = FALSE & (CEIL((y OF tmagv)/10) ≠ (x OF tmagv))
374
375         IF frombtob[ sbnr OF tmagv,agvberthnb OF tmagv] >0
376             frombtob[ sbnr OF tmagv,agvberthnb OF tmagv] ←frombtob→
377             [ sbnr OF tmagv,agvberthnb OF tmagv] -1
378             yyy OF tmagv←y OF tmagv
379             xxx OF tmagv←FLOOR((yyy OF tmagv-1)/20)
380             zzz OF tmagv←FLOOR((yyy OF tmagv-1)/20)
381             yyy OF tmagv←(y OF tmagv)-(zzz OF tmagv)*20
382             IF (yyy OF tmagv)MODULO(2)=0
383                 yyy OF tmagv ← yyy OF tmagv -1
384             END
385             @IF (yyy OF tmagv)> 10
386                 @yyy OF THIS agv← yyy OF tmagv -10
387             @END
388
389             yyy OF tmagv ← yyy OF tmagv + 20*(xxx OF tmagv)
390             tmcrane← FIRST crane IN cranelist WITH cranenumber = yyy OF tmagv
391             JOIN tmagv TO assignedqueue OF tmcrane
392
393             JOIN tmagv TO cranequeue OF tmcrane IN FRONT POSITION
394             REMOVE tmagv FROM agvworklist
395
396
397     WAIT
398         END
399         IF frombtob[ sbnr OF tmagv,agvberthnb OF tmagv] =0
400             IF frombtob[ agvberthnb OF tmagv,CEIL((y OF tmagv)/60)] =0
401                 JUMP TO yardshuffleallocation
402                 JUMP TO agv_stack_stack
403                 bye OF tmagv ← TRUE
404                 IF TMAGV IS ACTIVE
405                     CANCEL TMAGV
406                 END
407                 ACTIVATE tmagv FROM shuffling IN agvmod IF tmagv IS NOT ACTIVE
408                 REMOVE tmagv FROM agvworklist
409
410                 WAIT
411             END
412
413             frombtob[ agvberthnb OF tmagv,CEIL((y OF tmagv)/60)] ←frombtob→
414             [ agvberthnb OF tmagv,CEIL((y OF tmagv)/60)] -1
415             JUMP TO yardshuffleallocation
416             JUMP TO agv_stack_stack
417             IF TMAGV IS ACTIVE
418                 CANCEL TMAGV
419             END
420             ACTIVATE tmagv FROM shuffling IN agvmod IF tmagv IS NOT ACTIVE
421             REMOVE tmagv FROM agvworklist
422
423             WAIT
424
425         END
426     END
427
428     @@@@
```

```
435         IF frombtob[ agvberthnb OF tmagv,CEIL((y OF tmagv)/60)] =0
436             JUMP TO yardshuffleallocation
437             JUMP TO agv_stack_stack
438             bye OF tmagv ← TRUE
439             IF TMAGV IS ACTIVE
440                 CANCEL TMAGV
441             END
442             ACTIVATE tmagv FROM shuffling IN agvmod IF tmagv IS NOT ACTIVE
443             REMOVE tmagv FROM agvworklist
444
445         WAIT
446
447         END
448         JUMP TO yardshuffleallocation
449         JUMP TO agv_stack_stack
450
451
452         IF TMAGV IS ACTIVE
453             CANCEL TMAGV
454         END
455         ACTIVATE tmagv FROM shuffling IN agvmod IF tmagv IS NOT ACTIVE
456         REMOVE tmagv FROM agvworklist
457         WAIT
458
459     END
460
461 END
462 WAIT
463
```

The crane does not do anything until a agv is assigned or an agv enters a cranequeue

If length of cranequeue > 0 there is a rgv waiting under the crane **Line 8 to 25**

If **aantalagv** >0 it means a rgv has dropped its container on a service point. The rgv is gone and the crane recognises that a container has to be transported into the warehouse line 28 **readyqueue** is a set containing rgv's that have dropped a container onto a service point this does not mean the rgv still has to be in the warehouse

crane_cycletime is a macro which creates a cycle time for the crane

65 seconds together with the 15 seconds **line 31** are the working time for the crane to get the container into the right bay This time is deterministic in contrast to **line 36** just one of the two can be used

stackcount are the reserved service points in the stack
klaarcount is the counter that counts the containers that are placed on the service points by the crane, ready to be taken away by a rgv.
From line 48 to 59 If stackcount >=2 & klaarcount >=1 this means the crane has been anticipating and has already dropped a container onto a service point ready to be taken away by a rgv

```
 1 begin:
 2
 3 WAIT WHILE aantalagv OF THIS crane =0 & LENGTH OF assignedqueue OF THIS crane = 0
 4 & LENGTH OF craneladenqueue OF THIS crane =0 & LENGTH OF cranelossenqueue OF THIS
 5 & LENGTH OF cranequeue OF THIS crane = 0
 6
 7
 8 IF LENGTH OF cranequeue OF THIS crane > 0
 9     craneagv OF THIS crane ←FIRST AGV IN cranequeue OF THIS crane
10         IF craneagv OF THIS crane BELONGS TO assignedqueue OF THIS crane
11             assignedagv OF THIS crane ←craneagv OF THIS crane
12             REMOVE assignedagv OF THIS crane FROM assignedqueue OF THIS crane
13             @GOTO halverwege
14         END
15         IF loadedagv OF craneagv OF THIS crane = FALSE
16             REMOVE craneagv OF THIS crane FROM cranequeue OF THIS crane
17             JOIN craneagv OF THIS crane TO craneladenqueue OF THIS crane
18             GOTO begin
19         END
20         IF loadedagv OF craneagv = TRUE
21             REMOVE craneagv OF THIS crane FROM cranequeue OF THIS crane
22             JOIN craneagv OF THIS crane TO cranelossenqueue OF THIS crane
23             GOTO begin
24         END
25 END
26
27 IF aantalagv OF THIS crane >0
28         craneagv OF THIS crane ←FIRST OF readyqueue OF THIS crane
29         REMOVE craneagv OF THIS crane FROM readyqueue OF THIS crane
30         aantalagv OF THIS crane ← aantalagv OF THIS crane-1
31         WORK 15 SECONDS
32         @cranestack OF THIS crane ←FIRST stack IN stacklist WITH stacknumber = yy
33         cranestack OF THIS crane ←FIRST stack IN stacklist WITH stacknumber = rr (
34         stackcount OF cranestack OF THIS crane ←stackcount OF cranestack OF THIS
35         REMOVE craneagv FROM wantasqueue OF cranestack OF THIS crane IF craneagv
36          OF cranestack OF THIS crane
37         @JUMP TO CRANE_CYCLETIME            @HOREN BIJ HET EXPERIMENT
38         @JUMP TO crane_outportations
39         @WORK (F OF THIS CRANE -15)SECONDS  @hOREN BIJ HET EXPERIMENT
40         WORK 65 SECONDS
41         GOTO begin
42 END
43 IF LENGTH OF cranelossenqueue OF THIS crane  > 0
44     craneagv OF THIS crane ← FIRST agv IN cranelossenqueue OF THIS crane
45     cranestack OF THIS crane ← agvstack OF craneagv OF THIS crane
46     REMOVE craneagv FROM wantasqueue OF cranestack OF THIS crane IF craneagv BELO
47      TO wantasqueue OF cranestack OF THIS crane
48     IF shuffleagv OF craneagv = TRUE
49         IF stackcount OF cranestack OF THIS crane ≥2 & klaarcount OF THIS crane ≥
50             REMOVE craneagv FROM cranelossenqueue OF THIS crane
51             stackcount OF cranestack←stackcount OF cranestack -1
52             klaarcount OF THIS crane ←klaarcount OF THIS crane -1
53             rr OF craneagv←stacknumber OF cranestack OF THIS crane
54             JOIN craneagv TO readyqueue OF THIS crane
55             aantalagv OF THIS crane← aantalagv OF THIS crane +1
56             WAIT 15 SECONDS
57             JOIN craneagv OF THIS crane TO agvworklist
58             ACTIVATE craneagv FROM begin IN agvmod
59             GOTO final
60         END
61     END
62     GOTO volleagv
```

If length of craneladenqueue>0 **line 64 to 82** it means there is an empty in the stack that wants a container if stackcount >=2 & klaarcount >=1 there is a container ready for the rgv if not **goto line 81**

Assignedqueue is not empty, there is a rgv assigned **line 84 to 126**

If it is a shuffle rgv **line 87 to 106**
line 88 to 99 the crane gets a container for the assigned rgv and puts it on a service point this can be done because there is room at the service points

klaarcount +1 this means there is a container ready to be taken away

Line 100 to 105 there is no service point ready for the crane to put a container on. In this case the crane cannot do anything yet for the assigned rgv

line 107 the rgv is assigned but is not a shuffle rgv, this means it has to be a rgv that wants a container to load the vessel

line 110 to 119 the crane gets a container from the warehouse and puts it on a service point this can be done because there is room at the service points

The crane can not yet get a container because the service points are taken there is nothing the crane can do at the moment for this rgv

```
END
IF LENGTH OF craneladenqueue OF THIS crane >0
    craneagv OF THIS crane ←FIRST OF craneladenqueue OF THIS crane
    cranestack OF THIS crane ← agvstack OF craneagv OF THIS crane

    REMOVE craneagv FROM wantasqueue OF cranestack OF THIS crane IF craneagv→
     BELONGS TO wantasqueue OF cranestack OF THIS crane
    IF shuffleagv OF craneagv = TRUE
        IF stackcount OF cranestack ≥2 & klaarcount OF THIS crane ≥1
            REMOVE craneagv FROM craneladenqueue OF THIS crane
            stackcount OF cranestack←stackcount OF cranestack -2
            klaarcount OF THIS crane ←klaarcount OF THIS crane -1
            JOIN craneagv OF THIS crane TO agvworklist
            ACTIVATE craneagv FROM begin IN agvmod
            loadedagv OF craneagv←TRUE
            frombtob[ sbnr OF craneagv,agvberthnb OF craneagv] ← frombtob[ sbnr OF c
            GOTO final
        END
    END
    GOTO legeagv
END
IF assignedqueue OF THIS crane IS NOT EMPTY
    assignedagv OF THIS crane ← FIRST agv IN assignedqueue OF THIS crane
    halverwege:
    IF shuffleagv OF assignedagv OF THIS crane = TRUE
        cranestack OF THIS crane←FIRST stack IN stacklist WITH stacknumber = y OF
        REMOVE craneagv FROM wantasqueue OF cranestack OF THIS crane
        IF stackcount OF cranestack OF THIS crane ≤ 1
            stackcount OF cranestack OF THIS crane← stackcount OF cranestack OF T
            REMOVE assignedagv OF THIS crane FROM assignedqueue OF THIS crane
            @JUMP TO CRANE_CYCLETIME            @HOREN BIJ HET EXPERIMENT
            @JUMP TO crane_outportations
            @WORK (F OF THIS CRANE )SECONDS @hOREN BIJ HET EXPERIMENT
            WORK 80 SECONDS
            klaarcount OF THIS crane ← klaarcount OF THIS crane + 1
            GOTO begin
        END
        IF stackcount OF cranestack OF THIS crane >1
            REMOVE assignedagv FROM assignedqueue OF THIS crane
            JOIN assignedagv OF THIS crane TO wantasqueue OF cranestack OF THIS c
            ACTIVATE cranestack OF THIS crane  FROM begin IN stackmod IF cranesta
            GOTO begin
        END
    END
    IF shuffleagv OF assignedagv OF THIS crane = FALSE
        cranestack OF THIS crane ←FIRST stack IN stacklist WITH stacknumber = y O
        REMOVE craneagv FROM wantasqueue OF cranestack OF THIS crane IF craneagv
         wantasqueue OF cranestack OF THIS crane
        IF stackcount OF cranestack OF THIS crane≤2
            stackcount OF cranestack OF THIS crane ← stackcount OF cranestack OF
            REMOVE assignedagv OF THIS crane FROM assignedqueue OF THIS crane
            @JUMP TO CRANE_CYCLETIME            @HOREN BIJ HET EXPERIMENT
            @JUMP TO crane_outportations
            @WORK (F OF THIS CRANE )SECONDS @hOREN BIJ HET EXPERIMENT
            WORK 80 SECONDS
            klaarcount OF THIS cranE ← KLAARCOUNT OF THIS CRANE +1
            GOTO begin
        END
        IF stackcount OF cranestack OF THIS crane  > 2
            REMOVE assignedagv OF THIS crane FROM assignedqueue OF THIS crane
            JOIN assignedagv OF THIS crane TO wantasqueue OF cranestack OF THIS c
             IF ASSIGNEDAGV  BELONGS NOT TO WANTASQUEUE OF CRANESTACK OF THIS CRA
```

Loaded rgv arrives at the crane

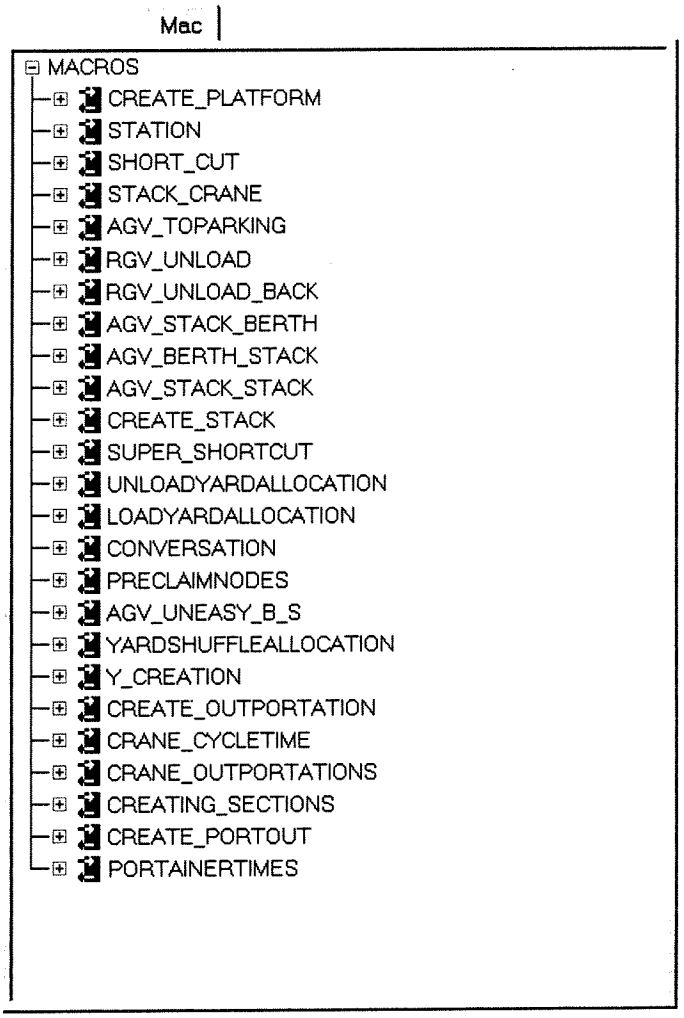
line 129 to 160 the crane serves the loaded agv

empty rgv arrives at the crane

```
125             GOTO begin
126         END
127     END
128 END
129 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
130 @loaded agv arrives at the crane@
131 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
132
133 volleagv:
134 craneagv OF THIS crane ← FIRST agv IN cranelossenqueue OF THIS crane
135 REMOVE craneagv FROM assignedqueue OF THIS crane IF craneagv BELONGS→
136      TO assignedqueue OF THIS crane
137 CANCEL craneagv  OF THIS crane IF craneagv OF THIS crane IS ACTIVE
138 IF loadedagv OF craneagv OF THIS crane = TRUE
139     loadedagv OF craneagv ← FALSE
140     REMOVE craneagv FROM cranelossenqueue→
141      OF THIS crane IF craneagv BELONGS TO cranelossenqueue OF THIS crane
142     IF x OF craneagv = 0
143         JOIN craneagv TO agvworklist
144         ACTIVATE craneagv  FROM begin IN agvmod
145         GOTO final
146     END
147     IF bye OF craneagv = TRUE
148         ACTIVATE craneagv FROM toparking IN agvmod
149         GOTO final
150     END
151     gozert:
152     @JUMP TO CRANE_CYCLETIME            @HOREN BIJ HET EXPERIMENT
153     @JUMP TO crane_outportations
154     @WORK (F OF THIS CRANE )SECONDS @hOREN BIJ HET EXPERIMENT
155     WORK 80 SECONDS
156     JOIN craneagv TO agvworklist
157     ACTIVATE craneagv FROM begin IN agvmod
158     GOTO final
159 END
160 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
161 @empty agv arrives at the crane  @
162 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
163 legeagv:
164 craneagv ← FIRST agv IN craneladenqueue OF THIS crane
165 REMOVE craneagv FROM assignedqueue OF THIS crane IF craneagv BELONGS→
166  TO assignedqueue OF THIS crane
167 IF loadedagv OF craneagv = FALSE
168
169     IF bye OF craneagv = TRUE & shuffleagv OF craneagv = TRUE
170         ACTIVATE craneagv FROM toparking IN agvmod
171         REMOVE craneagv OF THIS crane FROM craneladenqueue OF THIS crane
172         GOTO final
173     END
174     prima:
175     @JUMP TO CRANE_CYCLETIME            @HOREN BIJ HET EXPERIMENT
176     @JUMP TO crane_outportations
177     @WORK (F OF THIS CRANE )SECONDS     @hOREN BIJ HET EXPERIMENT
178     WORK 80  SECONDS
179     loadedagv OF craneagv OF THIS crane ← TRUE
180     @prima:
181
182     IF bye OF craneagv OF THIS crane = TRUE & shuffleagv OF craneagv = TRUE
183         ACTIVATE craneagv FROM toparking IN agvmod
184         REMOVE craneagv FROM craneladenqueue OF THIS crane
185         GOTO final
186     END
```

```
187
188
189
190     IF loadedagv OF craneagv OF THIS crane = FALSE
191         loadedagv OF craneagv OF THIS crane ← TRUE
192     END
193
194     REMOVE craneagv OF THIS crane FROM craneladenqueue OF THIS crane
195     JOIN craneagv OF THIS crane TO agvworklist
196     ACTIVATE craneagv OF THIS crane FROM begin IN agvmod
197     yy OF craneagv OF THIS crane ←y OF craneagv OF THIS crane
198     GOTO final
199 END
200 final:
201 GOTO begin
202
```

Appendix 28-III Macro's



Each portainerlocation has two platforms these are the actual lanes under the portainer
A platform consists of two sections a way in and a way out

n_in is a reference to a node, node_in the entrance of the platform

pl_x & pl_y are the x and y co-ordinates of the platform

Platforms is a set containing all the platforms

```
 1 THIS platform ← NEW platform CALLED char
 2 way_in ← NEW section
 3 way_out ← NEW section
 4 row ← NEW SET
 5 n_in ← READ FROM hconfig
 6 n_out ← READ FROM hconfig
 7 THIS NODE ← nodes[ n_in]
 8 outs[ 2] ← way_in
 9 side_track[ 1] ← station
10 pl_x ← READ FROM hconfig
11 pl_y ← READ FROM hconfig
12 end_point OF way_in ← THIS platform
13 s_length OF way_in ← distance_to(pl_x,pl_y)
14 THIS node ← nodes[ n_out]
15 end_point OF way_out ← THIS node
16 s_length OF way_out ← distance_to(pl_x,pl_y)
17 interval_check ← NEW POINTSTREAM
18 JOIN THIS platform TO platforms
19 RETURN
```

This is the time the rgv needs slow down and turn is wheels while it is on the node **line 6**

Line 5 &6 This is the time the rgv needs to get of the main track and pass the entrance node of the way_in
line 8 to 15 decide which exit to take. At the node there are two exits, because there are two way_ins for two portainerlocations. The right exit is decided by the x of the rgv
x is an attribute of the rgv referring to a portainerlocation

exittime is a specific time for a rgv to pull up and leave the node

At the moment the rgv has left the node it unclaims the node
s_length is the section length
this is the section way_in of the station

In line 22 to 33 the entrance node is given to the rgv it will need this node later when after the portainer it want to continue its journey, the entrance node is the end point of way out of the portainer.there are two different entrance nodes to be given depending on the x of the rgv

In line 35 and 36 the rgv has eventually reached the portainer and enters the portlocqueue waiting to be served

```
 1 @ a trolley enters a portainer to be @
 2 @ handled and to obtain a next task  @
 3 @-----------------------------------@
 4
 5 WORK (nodedistance1 OF cur_node)/(nodespeed OF cur_node)
 6 WAIT 4 SECONDS
 7
 8 IF (x OF THIS agv)MODULO(2)≠1
 9     cur_sec← outs[ 3] OF cur_node
10 END
11
12 IF (x OF THIS agv)MODULO(2)=1
13     cur_sec ← outs[ 2] OF cur_node
14 END
15
16 WORK exittime OF cur_node       @optrekken vanuit stand
17
18 claimed OF cur_node ← claimed OF cur_node - 1
19 WORK (s_length OF cur_sec/low_speed)+ 2*(low_speed/acc)
20
21
22 IF (x OF THIS agv)MODULO(2)≠1
23     cur_sec← outs[ 3] OF cur_node
24     cur_node←end_point OF cur_sec
25     JOIN cur_node TO entrance OF THIS agv IF entrance OF THIS agv IS EMPTY
26 END
27
28 IF (x OF THIS agv)MODULO(2)=1
29     stop ← end_point OF cur_sec
30     cur_sec←way_out OF stop
31     cur_node←end_point OF cur_sec
32     JOIN cur_node TO entrance OF THIS agv IF entrance OF THIS agv IS EMPTY
33 END
34
35 agvportloc←FIRST portloc IN portlocations WITH portlocnumber = x OF THIS agv
36 ENTER portlocqueue OF agvportloc
37
38
39
40
41 WAIT
42
43
44 RETURN
45
46
```

In **line 6 to 186** a rgv is at a node with two outs, outs[1] refers to the main track and outs[2] refers to the stack the rgv has to go to the stack

The rgv has to slow down in **line 13** and turn its wheels **line 14**

stack row a set containing all the rgv's that are in the stack at that moment

rgv pulls up
rgv declaims the node

Rgv drives into the stack

Line 30 to 42 keeps count of the stack

the rgv has to wait 15 seconds this is for the shuttle to drive onto the rgv
If line 32 counts this is a rgv unloading a vessel the rgv has dropped a container which eventually has to go to another berth to be taken away.

If line 35 counts this rgv is shuffling and has just delivered a container to the right stack
inplacecounter counts the containers that are in the right stack

If line 39 counts this rgv's shuffling and has just delivered a container to the right stack but not at the rgv's own berth.

In line 47 to 66 a loaded rgv enters a stack this rgv helps unloading a vessel

If stackcount > 3 all the service points are reserved
the rgv has to wait in the cranelossenqueue

If stackcount < 3 the rgv gets a service point to drop the container

The rgv drops its container at a service point aantalagv +1 this takes time but the 15 seconds are already deducted in **line 31**

The rgv is ready and wants a new task from the trafficmaster "enters the agvworklist"

```
@ The AGV arrives at an exit node. This exit node can be a @
@      regular exit node or an exeptional exit node.       @
@----------------------------------------------------------@


@--------------------------------------------------------------------------@
@The AGV arrives at a stack-entrance exeption node and identifies the node with@
@    3 exit possibilities, outs[ 1] , outs[ 2] , and decides to enter the stack    @
@--------------------------------------------------------------------------@

IF cur_node IS nodes[ 30*CEIL(y/20)+200-(y-11)MODULO(10)] & route OF THIS agv IS EM

    WORK (nodedistance1 OF cur_node)/(nodespeed OF cur_node)
    WORK 4 SECONDS @wieldraaings boete!?
    agvstack←FIRST stack IN stacklist WITH stacknumber = y OF THIS agv
    beeninstack OF THIS agv←TRUE
    ENTER stack_row OF agvstack OF THIS agv

    WORK exittime OF cur_node   @optrekken tot in het stack

    cur_sec ← outs[ 3]  OF cur_node

    WORK ((s_length OF cur_sec)/(2*low_speed))+ 2*(low_speed/acc)

    claimed OF cur_node ← claimed OF cur_node - 1
    agvstack←FIRST stack IN stacklist WITH stacknumber = y OF THIS agv

    jump to y creation

    agvcrane OF THIS agv← FIRST crane IN cranelist WITH cranenumber = yyy OF THIS
    WAIT 15 SECONDS
    IF shuffleagv OF THIS agv ≠ TRUE & loadedagv OF THIS agv = TRUE
        frombtob[ agvberthnb OF THIS agv, sbnr OF THIS agv] ←frombtob[ agvberthnb OF
    END
    IF shuffleagv OF THIS agv = TRUE & (sbnr OF THIS agv =0) & loadedagv OF THIS
        inplacecounter[ agvberthnb OF THIS agv] ← inplacecounter[ agvberthnb OF THIS

    END
    IF shuffleagv OF THIS agv = TRUE &(sbnr OF THIS agv) ≠0 & loadedagv OF THIS a
        inplacecounter[ sbnr OF THIS agv ] ← inplacecounter[ sbnr OF THIS agv] +1

    END
    REMOVE THIS agv FROM assignedqueue OF agvcrane IF THIS agv BELONGS TO assigne
    agvstack←FIRST stack IN stacklist WITH stacknumber = y OF THIS agv

    REMOVE THIS agv FROM wantasqueue OF agvstack IF THIS agv BELONGS TO wantasque
IF loadedagv OF THIS agv =TRUE & shuffleagv OF THIS agv = FALSE
    yard OF THIS agv ← TRUE
    IF stackcount OF agvstack ≥3
            ENTER cranelossenqueue OF agvcrane OF THIS agv
            WAIT
    END
    IF stackcount OF agvstack < 3
        stackcount OF agvstack ← stackcount OF agvstack + 1
        aantalagv OF agvcrane ← aantalagv OF agvcrane +1
        LEAVE agvworklist IF THIS agv BELONGS TO agvworklist
        loadedagv OF THIS agv←FALSE
        ENTER agvworklist
        yy OF THIS agv ← y OF THIS agv
         rr OF THIS agv← y OF THIS agv
        ENTER readyqueue OF agvcrane
        WAIT WHILE trafficmaster IS ACTIVE
```

If these conditions count the rgv is helping to load a vessel **line 68 to 90**

if stackcount >=1 & klaarcount >=1 there is a container waiting at a service point to be taken away by a rgv this happens in **line 71 to 81**
line 72 to 76 the properties of the stack and the rgv are adjusted and the rgv enters the rgvworklist to attain a next assignment

in **line 82 to 89** there is no container ready for the rgv, it has to wait in the craneladenqueue

Line 94 to 181 a shuffle rgv enters the stack

In **line 97 to 118** there is a container waiting for the rgv

If the rgv is not loaded it will not nead a service point to drop its container so this service point can be used for other purposes

If the condition **in line 109** counts it means a rgv will deliver a container but there is no shuffle container to take with him, it will leave the stack empty
aantal +1 means a container is dropped on a service point

In **line 119 to 134** a rgv enters its own stack
Own stack is a stack behind the portainer the rgv was working for in the beginning
In **line 120 to 125** the rgv enters empty

nietsterug is **line 168** in this line the rgv enters the agvworklist to ask the trafficmaster if he has to get another container from this stack.

```
63          ACTIVATE trafficmaster FROM begin IN trafficmastmod IF trafficmaster IS N(
64          WAIT
65      END
66  END
67
68  IF loadedagv OF THIS agv = FALSE & shuffleagv OF THIS agv =FALSE
69      yard OF THIS agv ← TRUE
70      agvstack←FIRST stack IN stacklist WITH stacknumber = y OF THIS agv
71      IF stackcount OF agvstack ≥1 & klaarcount OF agvcrane OF THIS agv ≥1
72          klaarcount OF agvcrane←klaarcount OF agvcrane -1
73          stackcount OF agvstack ← stackcount OF agvstack -1
74          loadedagv OF THIS agv ← TRUE
75          yard OF THIS agv ← TRUE
76          yy OF THIS agv ← y OF THIS agv
77          ENTER agvworklist
78          WAIT WHILE trafficmaster IS ACTIVE
79          ACTIVATE trafficmaster FROM begin IN trafficmastmod IF trafficmaster IS N(
80          WAIT
81      END
82      IF stackcount OF agvstack =0
83          ENTER craneladenqueue OF agvcrane OF THIS agv
84          WAIT
85      END
86      IF klaarcount OF agvcrane =0
87          ENTER craneladenqueue OF agvcrane OF THIS agv
88          WAIT
89      END
90  END
91
92
93  IF shuffleagv OF THIS agv =TRUE
94      yard OF THIS agv ← TRUE
95      agvstack←FIRST stack IN stacklist WITH stacknumber = y OF THIS agv
96      IF stackcount OF agvstack ≥ 2 & KLAARCOUNT OF AGVCRANE OF THIS agv≥1
97          stackcount OF agvstack ← stackcount OF agvstack -1
98          IF agvberthnb OF THIS agv = (CEIL(stacknumber OF agvstack OF THIS agv /60
99              & sbnr OF THIS agv ≠ 0
100             IF loadedagv OF THIS agv = FALSE & frombtob[ agvberthnb OF THIS agv,sb:
101                 stackcount OF agvstack ← stackcount OF agvstack -1
102                 GOTO fromhere
103             END
104             IF loadedagv OF THIS agv = FALSE & frombtob[ agvberthnb OF THIS agv,sb:
105                 stackcount OF agvstack ← stackcount OF agvstack -1
106                 GOTO leegin
107             END
108
109             IF frombtob[ agvberthnb OF THIS agv,sbnr OF THIS agv] ≤0
110                 JOIN THIS agv TO readyqueue OF agvcrane
111                 aantalagv OF agvcrane OF THIS agv ← aantalagv OF agvcrane OF THIS
112                 fromhere:
113                 rr OF THIS agv← y OF THIS agv
114                 yy OF THIS agv ← y OF THIS agv
115                 loadedagv OF THIS agv ← FALSE
116                 GOTO nietsterug
117             END
118         END
119         IF agvberthnb OF THIS agv = (CEIL(stacknumber OF agvstack OF THIS agv /60
120             IF loadedagv OF THIS agv = FALSE
121                 stackcount OF agvstack ← stackcount OF agvstack -1
122                 rr OF THIS agv← y OF THIS agv
123                 yy OF THIS agv ← y OF THIS agv
124                 GOTO nietsterug
```

line 126 to 134 if loaded is true the container is dropped on a service point

goto nietsterug **line 167** in this line the rgv enters the agvworklist to ask the trafficmaster if he has to get another container from this stack.

In l**ine 136 to 155** the shuffle agv enters a stack but not its own stack this is the same as line **100 to 132**

The moment the rgv passes **line 156** it means it is about to receive a container from the stack These are the 20 seconds in **line 156**. The shuttle puts the container on the rgv

the rgv enters the agvworklist, requesting the agvmaster for work

```
125             END
126             IF loadedagv OF THIS agv = TRUE
127                 JOIN THIS agv TO readyqueue OF agvcrane
128                 aantalagv OF agvcrane OF THIS agv ← aantalagv OF agvcrane OF THIS
129                 rr OF THIS agv← y OF THIS agv
130                 yy OF THIS agv ← y OF THIS agv
131                 loadedagv OF THIS agv ←FALSE
132                 GOTO nietsterug
133             END
134         END
135
136         IF agvberthnb OF THIS agv ≠ (CEIL(stacknumber OF agvstack OF THIS agv /60
137             IF loadedagv OF THIS agv = FALSE & frombtob[ sbnr OF THIS agv,agvberth:
138                 stackcount OF agvstack ← stackcount OF agvstack -1
139                 GOTO fromthere
140             END
141             IF loadedagv OF THIS agv = FALSE & frombtob[ sbnr OF THIS agv,agvberth:
142                 stackcount OF agvstack ← stackcount OF agvstack -1
143                 GOTO leegin
144             END
145
146             IF frombtob[ sbnr OF THIS agv,agvberthnb OF THIS agv] ≤0
147                 JOIN THIS agv TO readyqueue OF agvcrane
148                 aantalagv OF agvcrane OF THIS agv ← aantalagv OF agvcrane OF THIS
149                 fromthere:
150                 rr OF THIS agv← y OF THIS agv
151                 yy OF THIS agv ← y OF THIS agv
152                 loadedagv OF THIS agv ← FALSE
153                 GOTO nietsterug
154             END
155         END
156         WAIT 20 SECONDS
157         frombtob[ sbnr OF THIS agv, agvberthnb OF THIS agv] ← frombtob[ sbnr OF THI:
158         @ nu moet hij al de juiste eigenschappen meekrijgen
159         rr OF THIS agv← y OF THIS agv
160         JOIN THIS agv TO readyqueue OF agvcrane
161
162         aantalagv OF agvcrane OF THIS agv ← aantalagv OF agvcrane OF THIS agv +1
163         leegin:
164         klaarcount OF agvcrane OF THIS agv← klaarcount OF agvcrane OF THIS agv -1
165         loadedagv OF THIS agv ← TRUE
166         yy OF THIS agv ← y OF THIS agv
167         nietsterug:
168         ENTER agvworklist
169         WAIT WHILE trafficmaster IS ACTIVE
170         ACTIVATE trafficmaster FROM begin IN trafficmastmod IF trafficmaster IS N(
171         WAIT
172     END
173     IF stackcount OF agvstack OF THIS agv≤1
174         IF loadedagv OF THIS agv = FALSE & stackcount OF agvstack OF THIS agv =1
175             klaarcount OF agvcrane← klaarcount OF agvcrane -1
176             GOTO leegin
177         END
178         ENTER cranequeue OF agvcrane
179         WAIT
180     END
181 END
182 ENTER cranequeue OF agvcrane
183 WAIT
184 RETURN
185 END
186
```

In **line 187 to 225** the rgv arrives at a shortcut exception behind a portainer and chooses to go right (examples of these nodes are node 250, 280, 310....)

if the condition in **line 191** counts it means the rgv can choose between to portainerlocations and chooses to take the left one as a result of his x value, the second condition **line 192** is the slowing down of the rgv at the exit node
The 4 seconds are the rotation of the wheels

exittime is a specific time for a rgv to pull up and leave the node
outs[3] refers to the left side

rgv drives the length of the shortcut

Line 203 and 204 are the nodes that have to be unclaimed for the rgv to enter the main track again

in **line 206 to 214** the rgv waits **[208]** until the nodes are not claimed the moment they are not claimed the rgv starts driving **[211]**

the rgv activates claim 2 in the nodemod. This is so the node can declaim itself normally this is done by the rgv but this is one of the exceptions

4 seconds is the discrotation of the rgv

In **line 225 to 261** the same happens as in **line 187 to 224** except that the rgv takes the left exit

```
187 @-------------------------------------------------------@
188 @AGV arrives at a shortcut-exeption behind the portainer@
189 @     coming from the stack and chooses to go left      @
190 @-------------------------------------------------------@
191 IF cur_node IS nodes[ 181+(CEIL((x+0.5)/2))*30]  &  (x OF THIS agv)MODULO(2)≠1
192     WORK (nodedistance1 OF cur_node)/(nodespeed OF cur_node)     @Extra remboete o
193                                                                  @tot stilstand te kom
194     WORK 4 SECONDS @wieldraaings boete!?
195
196     WORK exittime OF cur_node
197     cur_sec ← outs[ 3]  OF cur_node
198     claimed OF cur_node ← claimed OF cur_node - 1
199
200     WORK MAX(0,(((s_length OF cur_sec)/(low_speed))-2*(exittime OF cur_node)))
201     cur_node ← end_point OF cur_sec
202
203     claim1 ← FIRST node IN node_list WITH node_number = (node_number OF cur_node
204     claim2 ← FIRST node IN node_list WITH node_number = (node_number OF cur_node
205
206     IF claimed OF cur_node > 0 | claimed OF claim1>0 | claimed OF claim2>0
207         WORK 2*(exittime OF cur_node)    @remmen(*1extra)
208         WAIT WHILE claimed OF cur_node >0 | claimed OF claim1>0 | claimed OF clai
209         ACTIVATE claim2 FROM special IN nodemod
210         claimed OF cur_node←claimed OF cur_node + 1
211         WORK 2*(exittime OF cur_node)    @optrekken(*2)
212         WORK 4 SECONDS
213         GOTO beter
214     END
215
216     claimed OF cur_node←claimed OF cur_node + 1
217     WORK exittime OF cur_node
218     WORK 4 SECONDS
219     WORK (nodedistance1 OF cur_node)/(nodespeed OF cur_node)
220
221
222     GOTO beter
223 END
224
225 @-------------------------------------------------------@
226 @AGV arrives at a shortcut-exeption behind the portainer@
227 @     coming from the stack and chooses to go right     @
228 @-------------------------------------------------------@
229 IF cur_node IS nodes[ 181+(CEIL((x+0.5)/2))*30]  &  (x OF THIS agv)MODULO(2)≠0
230     WORK (nodedistance1 OF cur_node)/(nodespeed OF cur_node)     @Extra remboete o
231                                                                  @tot stilstand te kom
232     WORK 4 SECONDS @wieldraaings boete!?
233
234     WORK exittime OF cur_node
235     cur_sec ← outs[ 2]  OF cur_node
236     claimed OF cur_node ← claimed OF cur_node - 1
237
238     WORK MAX(0,(((s_length OF cur_sec)/(low_speed))-2*(exittime OF cur_node)))
239     cur_node ← end_point OF cur_sec
240
241     claim1 ← FIRST node IN node_list WITH node_number = (node_number OF cur_node
242     claim2 ← FIRST node IN node_list WITH node_number = (node_number OF cur_node
243
244     IF claimed OF cur_node > 0 | claimed OF claim1>0 | claimed OF claim2>0
245         WORK 2*(exittime OF cur_node)    @remmen(*1extra)
246         WAIT WHILE claimed OF cur_node >0 | claimed OF claim1>0 | claimed OF clai
247         ACTIVATE claim2 FROM special IN nodemod
248         claimed OF cur_node←claimed OF cur_node + 1
```

rgv arrives at a regular exit node and the exit requires a 90 degree bend. **Line 262 to 313**
This is a bit different than when a rgv doesn't have to make a turn

```
249         WORK 2*(exittime OF cur_node)   @optrekken(*2)
250         WORK 8 SECONDS
251         GOTO beter
252     END
253
254     claimed OF cur_node←claimed OF cur_node + 1
255     WORK exittime OF cur_node
256     WORK 4 SECONDS
257     WORK (nodedistance1 OF cur_node)/(nodespeed OF cur_node)
258
259     GOTO beter
260 END
261
262 @-------------------------------------------------------------------------
263 @  AGV arrives at a regular exit node only the exit node requires 90 degree bend
264 @-------------------------------------------------------------------------
265 IF (node_number OF cur_node OF THIS agv)-(node_number OF lastnode OF THIS agv) =
266     IF ((((node_number OF cur_node)-200)MODULO(30)=0 )|(((node_number OF cur_node
267     & (node_number OF cur_node)<484
268         GOTO skip
269     END
270     WORK (nodedistance1 OF cur_node)/(nodespeed OF cur_node)     @Extra remboete o
271                                                                  @tot stilstand te kom
272     WORK 4 SECONDS  @wieldraaings boete!?
273
274     WORK exittime OF cur_node
275     cur_sec ← outs[ 2] OF cur_node
276     claimed OF cur_node ← claimed OF cur_node - 1
277
278     ENTER sec_row OF cur_sec
279
280     reference_node ← FIRST node IN node_list WITH node_number = 202
281     WORK MAX(0,((s_length OF cur_sec/(nodespeed OF reference_node))-2*(exittime O
282     cur_node ← end_point OF cur_sec
283
284     IF claimed OF cur_node > 0
285         @ hier gaat het gebeuren
286         IF LENGTH OF route OF THIS agv = 1
287         n←node_number OF exit OF THIS agv
288         exit ← FIRST node IN route OF THIS agv
289         n←node_number OF cur_node OF THIS agv
290             IF node_number OF cur_node OF THIS agv = node_number OF exit OF THIS
291             & (n = 218 | n=248 | n=278 | n=308 | n=338 | n=368| n=398| n=428| n=4
292             LEAVE sec_row OF cur_sec
293             cur_node OF THIS agv ← FIRST node IN route OF THIS agv
294             REMOVE FIRST node IN route OF THIS agv  FROM route OF THIS agv
295             weird OF THIS agv ← TRUE
296             RETURN
297             END
298         END
299         WORK 2*(exittime OF cur_node) @remmen(*1extra)
300         WAIT WHILE claimed OF cur_node >0
301         claimed OF cur_node←claimed OF cur_node + 1
302         WORK 2*(exittime OF cur_node) @optrekken(*2)
303         LEAVE sec_row OF cur_sec
304         GOTO beter
305     END
306
307     claimed OF cur_node←claimed OF cur_node + 1
308     WORK exittime OF cur_node
309
310     LEAVE sec_row OF cur_sec
```

The last possible option is a rgv takes a shortcut but doesn't have to take a turn

```
311     GOTO beter
312 END
313 skip:
314
315 @-------------------------------------------------------------------------
316 @   The only possibility that remains is that the agv exits without 90 dgr. turn
317 @-------------------------------------------------------------------------
318
319 reference_node ← FIRST node IN node_list WITH node_number = 202
320 WORK (nodedistance1 OF reference_node)/(nodespeed OF reference_node)
321
322 cur_sec ← outs[ 2] OF cur_node
323 claimed OF cur_node ← claimed OF cur_node - 1
324
325 ENTER sec_row OF cur_sec
326
327 WORK MAX(0,((s_length OF cur_sec/(nodespeed OF reference_node))-2*(nodedistance1 (
328 cur_node ← end_point OF cur_sec
329     IF claimed OF cur_node > 0
330
331     @ hier gaat het gebeuren
332         IF LENGTH OF route OF THIS agv = 1
333         n←node_number OF exit OF THIS agv
334         exit ← FIRST node IN route OF THIS agv
335         n←node_number OF cur_node OF THIS agv
336             IF node_number OF cur_node OF THIS agv = node_number OF exit OF THIS .
337             & (n = 218 | n=248 | n=278 | n=308 | n=338 | n=368| n=398| n=428| n=4
338             LEAVE sec_row OF cur_sec
339             cur_node OF THIS agv ← FIRST node IN route OF THIS agv
340             REMOVE FIRST node IN route OF THIS agv  FROM route OF THIS agv
341             weird OF THIS agv ← TRUE
342             RETURN
343             END
344         END
345     WORK 2*(exittime OF reference_node)      @remmen(*1extra)
346     WAIT WHILE claimed OF cur_node >0
347     claimed OF cur_node←claimed OF cur_node + 1
348     WORK 2*(exittime OF reference_node) @optrekken(*2)
349     LEAVE sec_row OF cur_sec
350     GOTO beter
351 END
352
353 claimed OF cur_node←claimed OF cur_node + 1
354 WORK (nodedistance1 OF reference_node)/(nodespeed OF reference_node)
355
356 @--AGV has to make a 90 drg. turn to get on the fastlane--@
357
358 exit ← FIRST node IN route OF THIS agv
359 IF cur_node IS NOT exit & ((node_number OF cur_node)<200 | (node_number OF cur_no
360
361
362
363     WORK (nodedistance1 OF reference_node)/(nodespeed OF reference_node)@!!!
364     WORK 4 SECONDS
365     WORK (nodedistance1 OF cur_node)/(nodespeed OF cur_node)
366 END
367
368 LEAVE sec_row OF cur_sec
369 beter:
370
371 exit ← FIRST node IN route OF THIS agv
372 REMOVE FIRST node IN route OF THIS agv  FROM route OF THIS agv
```

```
373 final:
374 RETURN
375
376
377
378
```

The stacking crane macro works the same as line **1 to 180** of the short cut.

```
 1 @-----------------------------------@
 2 @ a trolley enters a stack to be    @
 3 @ handled and to obtain a next task @
 4 @-----------------------------------@
 5
 6 WORK (nodedistance1 OF cur_node)/(nodespeed OF cur_node)
 7 WORK 8 SECONDS
 8 agvstack←FIRST stack IN stacklist WITH stacknumber = y OF THIS agv
 9 beeninstack OF THIS agv←TRUE
10 ENTER stack_row OF agvstack OF THIS agv
11
12 WORK exittime OF cur_node
13
14 cur_sec ← outs[ 2] OF cur_node
15
16 WORK (s_length OF cur_sec/low_speed)+ 2*(low_speed/acc)      @!!!!!
17
18 claimed OF cur_node ← claimed OF cur_node - 1
19
20 Jump to y creation
21
22 agvcrane← FIRST crane IN cranelist WITH cranenumber = yyy OF THIS agv
23
24 WAIT 15 SECONDS
25     IF shuffleagv OF THIS agv ≠ TRUE & loadedagv OF THIS agv = TRUE
26         frombtob[ agvberthnb OF THIS agv, sbnr OF THIS agv] ←frombtob[ agvberthnb OF
27     END
28     IF shuffleagv OF THIS agv = TRUE & (sbnr OF THIS agv =0) & loadedagv OF THIS
29         inplacecounter[ agvberthnb OF THIS agv] ← inplacecounter[ agvberthnb OF THIS
30     END
31
32     IF shuffleagv OF THIS agv = TRUE &(sbnr OF THIS agv) ≠0 & loadedagv OF THIS a
33         inplacecounter[ sbnr OF THIS agv ] ← inplacecounter[ sbnr OF THIS agv] +1
34     END
35
36
37 REMOVE THIS agv FROM assignedqueue OF agvcrane OF THIS agv IF THIS agv BELONGS TO
38 agvstack OF THIS agv←FIRST stack IN stacklist WITH stacknumber = y OF THIS agv
39 REMOVE THIS agv FROM wantasqueue OF agvstack OF THIS agv IF THIS agv BELONGS TO w
40
41 IF loadedagv OF THIS agv =TRUE & shuffleagv OF THIS agv = FALSE
42     yard OF THIS agv ← TRUE
43     IF stackcount OF agvstack ≥3
44             ENTER cranelossenqueue OF agvcrane OF THIS agv
45             WAIT
46     END
47     IF stackcount OF agvstack < 3
48         stackcount OF agvstack ← stackcount OF agvstack + 1
49         aantalagv OF agvcrane ← aantalagv OF agvcrane +1
50         LEAVE agvworklist IF THIS agv BELONGS TO agvworklist
51         loadedagv OF THIS agv←FALSE
52         yy OF THIS agv ← y OF THIS agv
53         ENTER agvworklist
54
55         rr OF THIS agv ← y OF THIS agv
56         ENTER readyqueue OF agvcrane
57         WAIT WHILE trafficmaster IS ACTIVE
58         ACTIVATE trafficmaster FROM begin IN trafficmastmod IF trafficmaster IS N
59         WAIT
60     END
61 END
62
```

```
IF loadedagv OF THIS agv = FALSE & shuffleagv OF THIS agv =FALSE
    yard OF THIS agv ← TRUE
    agvstack←FIRST stack IN stacklist WITH stacknumber = y OF THIS agv
    IF stackcount OF agvstack ≥1 & klaarcount OF agvcrane OF THIS agv ≥1
        klaarcount OF agvcrane←klaarcount OF agvcrane -1
        stackcount OF agvstack ← stackcount OF agvstack -1
        loadedagv OF THIS agv ← TRUE
        yy OF THIS agv ← y OF THIS agv
        ENTER agvworklist
        WAIT WHILE trafficmaster IS ACTIVE
        ACTIVATE trafficmaster FROM begin IN trafficmastmod IF trafficmaster IS N
        WAIT
    END
    IF stackcount OF agvstack =0
        ENTER craneladenqueue OF agvcrane OF THIS agv
        WAIT
    END
    IF klaarcount OF agvcrane =0
        ENTER craneladenqueue OF agvcrane OF THIS agv
        WAIT
    END
END

IF shuffleagv OF THIS agv =TRUE
    yard OF THIS agv ← TRUE
    agvstack←FIRST stack IN stacklist WITH stacknumber = y OF THIS agv
    IF stackcount OF agvstack ≥ 2 & KLAARCOUNT OF AGVCRANE OF THIS agv≥1
        stackcount OF agvstack ← stackcount OF agvstack -1
        IF agvberthnb OF THIS agv = (CEIL(stacknumber OF agvstack OF THIS agv /60
            IF loadedagv OF THIS agv = FALSE & frombtob[ agvberthnb OF THIS agv,sb
                stackcount OF agvstack ← stackcount OF agvstack -1
                GOTO fromhere
            END
            IF loadedagv OF THIS agv = FALSE & frombtob[ agvberthnb OF THIS agv,sb
                stackcount OF agvstack ← stackcount OF agvstack -1
                GOTO leegin
            END

            IF frombtob[ agvberthnb OF THIS agv,sbnr OF THIS agv] ≤0
                JOIN THIS agv TO readyqueue OF agvcrane
                aantalagv OF agvcrane OF THIS agv ← aantalagv OF agvcrane OF THIS
                fromhere:
                rr OF THIS agv← y OF THIS agv
                yy OF THIS agv ← y OF THIS agv
                loadedagv OF THIS agv ←FALSE
                GOTO nietsterug
            END
        END
        IF agvberthnb OF THIS agv = (CEIL(stacknumber OF agvstack OF THIS agv /60
            & sbnr OF THIS agv = 0
            IF loadedagv OF THIS agv = FALSE
                stackcount OF agvstack ← stackcount OF agvstack -1
                rr OF THIS agv← y OF THIS agv
                yy OF THIS agv ← y OF THIS agv
                GOTO nietsterug
            END
            IF loadedagv OF THIS agv = TRUE
                JOIN THIS agv TO readyqueue OF agvcrane
                aantalagv OF agvcrane OF THIS agv ← aantalagv OF agvcrane OF THIS
                rr OF THIS agv← y OF THIS agv
                yy OF THIS agv ← y OF THIS agv
                loadedagv OF THIS agv ←FALSE
```

```
125                 GOTO nietsterug
126             END
127         END
128
129         IF agvberthnb OF THIS agv ≠ (CEIL(stacknumber OF agvstack OF THIS agv /60
130             IF loadedagv OF THIS agv = FALSE & frombtob[ sbnr OF THIS agv,agvberth:
131                 stackcount OF agvstack ← stackcount OF agvstack -1
132                 GOTO fromthere
133             END
134             IF loadedagv OF THIS agv = FALSE & frombtob[ sbnr OF THIS agv,agvberth:
135                 stackcount OF agvstack ← stackcount OF agvstack -1
136                 GOTO leegin
137             END
138         IF frombtob[ sbnr OF THIS agv,agvberthnb OF THIS agv] ≤0
139                 JOIN THIS agv TO readyqueue OF agvcrane
140                 aantalagv OF agvcrane OF THIS agv ← aantalagv OF agvcrane OF THIS
141                 fromthere:
142                 rr OF THIS agv← y OF THIS agv
143                 yy OF THIS agv ← y OF THIS agv
144                 loadedagv OF THIS agv ←FALSE
145                 GOTO nietsterug
146             END
147         END
148         WAIT 20 SECONDS
149         frombtob[ sbnr OF THIS agv, agvberthnb OF THIS agv] ← frombtob[ sbnr OF THI:
150         @ nu moet hij al de juiste eigenschappen meekrijgen
151         rr OF THIS agv← y OF THIS agv
152         JOIN THIS agv TO readyqueue OF agvcrane
153
154         aantalagv OF agvcrane OF THIS agv ← aantalagv OF agvcrane OF THIS agv +1
155         leegin:
156         klaarcount OF agvcrane OF THIS agv← klaarcount OF agvcrane OF THIS agv -1
157         loadedagv OF THIS agv ← TRUE
158         yy OF THIS agv ← y OF THIS agv
159         nietsterug:
160         ENTER agvworklist
161         WAIT WHILE trafficmaster IS ACTIVE
162         ACTIVATE trafficmaster FROM begin IN trafficmastmod IF trafficmaster IS N(
163         WAIT
164     END
165     IF stackcount OF AGVSTACK OF THIS AGV≤1
166         IF loadedagv OF THIS agv = FALSE & stackcount OF AGVSTACK OF THIS AGV =1
167             stackcount OF agvstack ← stackcount OF agvstack -1
168             GOTO leegin
169         END
170         ENTER cranequeue OF agvcrane
171         WAIT
172     END
173 END
174 ENTER cranequeue OF agvcrane
175 WAIT
176 final:
177 RETURN
```

If the rgv was in a stack_row of a stack it is removed, normally it removes itself from a stack row if it starts driving after the stack. But in this case the rgv just disappears from the stack

If the rgv was a shuffling for a vessel is now removed from that shufflelist

```
 1 @agv to parking@
 2 tmx ← x OF THIS agv
 3 tmy ← y OF THIS agv
 4 agvstack←FIRST stack IN stacklist WITH stacknumber = tmy
 5 LEAVE stack_row OF agvstack OF THIS agv IF THIS agv BELONGS TO stack_row→
 6  OF agvstack OF THIS agv
 7 JOIN THIS agv TO agvparking IF THIS agv BELONGS NOT TO agvparking
 8
 9 bye OF THIS agv  ← FALSE
10 y OF THIS agv ← CEIL((tmx)/2)*20-18→
11 + ((-1 + tmx)MODULO(2))*9
12
13 yard OF THIS agv ← TRUE
14 IF entrance OF THIS agv IS NOT EMPTY
15     REMOVE EACH node IN entrance OF THIS agv FROM entrance OF THIS agv
16 END
17 REMOVE THIS agv FROM agvberthshufflelist[ agvberthnb OF THIS agv] →
18 IF THIS agv BELONGS TO agvberthshufflelist[ agvberthnb OF THIS agv]
19
20 REMOVE THIS agv FROM agvworklist IF THIS agv BELONGS TO agvworklist
21 REMOVE THIS agv FROM activeagvlist IF THIS agv BELONGS TO activeagvlist
22
23     FOR i ← 1 TO 10
24         tmport ← FIRST portainer IN quayrail WITH portainernumber = i
25         REMOVE THIS agv FROM  agvppactive OF tmport IF THIS agv BELONGS→
26          TO agvppactive OF tmport
27     END
28 WAIT
29
30 REMOVE THIS agv FROM agvppactiver OF tmport IF THIS agv BELONGS TO agvppactiver O
31 WAIT
```

agv_unload is a macro that creates the route from the portainer to the right stack
there are two possibilities depending on the portainerlocation the rgv has just left

line 7 to 11 if the portainerlocation is odd this condition counts

line 12 to 15 if the portainerlocation is even this condition counts

```
 1 begin:
 2
 3 tmx ← x OF tmagv
 4 tmy ← y OF tmagv
 5
 6
 7 IF (tmx)MODULO(2)=1
 8     @cur_node
 9     goede_node←nodes[ 30*CEIL(tmy/20)+200-(tmy-1)MODULO(10)+(tmx-1)MODULO(2)]
10     JOIN goede_node TO route OF tmagv
11 END
12 IF (tmx)MODULO(2)=0
13     goede_node←nodes[ 30*CEIL(tmy/20)+200-(tmy-11)MODULO(10)]
14     JOIN goede_node TO route OF tmagv
15 END
16 RETURN
```

This macro creates the route from the stack back to the portainer there are two possibilities
first in **line 6 to 9** the entrance node is given, this is the end node of the stack

line 10 to 14 is for the odd numbered portainerlocation

line 15 to 19 is for the even numbered portainerlocation

```
egin:

yy OF tmagv← y OF tmagv
tmx ← x OF tmagv

goede_stack←FIRST stack IN stacklist WITH stacknumber = yy OF tmagv
goede_section ←way_out_stack OF goede_stack @agvstack
goede_node ←end_point OF goede_section @agv_section
JOIN goede_node  TO entrance OF tmagv IF entrance OF tmagv IS EMPTY

IF (tmx)MODULO(2)=1
    goede_node ←nodes[ 212+CEIL((tmx-1)/2)*30]
    JOIN goede_node TO route OF tmagv
END
IF (tmx)MODULO(2)=0
    goede_node←nodes[ 212+CEIL((tmx)/2)*30]
    JOIN goede_node TO route OF tmagv
END

RETURN
```

agv_stack_berth is a macro that creates the route to get from a stack to a berth

First in **line 5 to 8** the entrance node is given to the rgv

if the statement in **line 11** counts the route has to be the same as expected in agv_lossen

A decision is made to let the odd rgv travel using the top route of the terminal **line 22 to 187** and the even rgv use the bottom route of the terminal **line 187 to the end**
If the condition in **line 22** counts the rgvnumber of the rgv is odd
The rgv has to leave the main route connecting the stacks this rgv is odd so it has to leave the stacks at the top **line 25 and 26** create this node

the rgv has to go back (left) this is possible without first crossing another lane
just one node is required to tell the rgv where to leave this main track this is created in **line 29**

the rgv has to go foreword (right), first the rgv has to cross the driving lane that is directed to the left this extra short cut is given in **line 35 and 36** and **line 37** creates the node where the rgv should leave the main track again. Thereafter It has to cross the other lane again These nodes could be (node260 line25, node116 line 35, node 85 if x is 4 line 37, node 114 **line 39**)

In **line 39** the rgv has to cross a node again

This node in **line 43** refers to the portainer location the rgv has to take, tmx refers to the portainer location

If the condition in **line 48** counts the rgv is located in a stack behind berth 1 or behind berth 3 and has to go to the middle berth, berth 2.
The rgv has to take the top route this node is the exit of the stack area **line 50** (for example if the rgv is located in stack 143 this node will be 440)

Line 53 if the rgv is located behind berth 3 the rgv will have to go back from berth 3 to 2
This is explained in **line 53 to 56** In **line 54** the exit node is given.

If the rgv is located behind berth 1 the rgv will have to go foreword to berth 2 (right). It already has the node from **line 50 and 51** to exit the stack area but because it has to go to the right it first has to cross a lane, this in contrast to the rgv that has to go left (**line 53 to 56**)
Line 60 and 61 is the crossing of the lane
Line 62 is the exit from the top main route

```
 1
 2 tmx ← x OF tmagv
 3 tmy ← y OF tmagv
 4
 5 goede_stack  ←FIRST stack IN stacklist WITH stacknumber = yy OF tmagv
 6 goede_section←way_out_stack OF goede_stack
 7 goede_node←end_point OF goede_section
 8 JOIN goede_node TO entrance OF tmagv IF entrance OF tmagv IS EMPTY
 9
10
11 IF easyload OF tmagv = TRUE
12
13     goede_node←nodes[ 212+CEIL((tmx-1)/2)*30]
14     JOIN goede_node TO route OF tmagv
15
16     GOTO later
17 END
18
19
20 begin:
21
22 IF (agvnumber OF tmagv)MODULO(2)=1| (AGVNUMBER OF TMAGV)MODULO(2)=0
23     IF tmx=CEIL(tmy/10)
24
25     goede_node ← nodes[ CEIL(tmy/20)*30+200]
26     JOIN goede_node TO route OF tmagv
27
28         IF (tmx)MODULO(2)=1            @de agv moet terug
29         goede_node ← nodes[ 118-(CEIL(CEIL((tmx+0.5)/2)-1))*2]
30         JOIN goede_node TO route OF tmagv
31         END
32
33         IF (tmx)MODULO(2)=0            @de agv moet heen
34         @goede_node ← nodes[ 119-CEIL(tmy/20)*2] @eerst eentje extra oversteken
35         goede_node ← end_point OF outs[ 2] OF goede_node
36         JOIN goede_node TO route OF tmagv
37         goede_node ← nodes[ CEIL(CEIL((tmx+0.5)/2))*2+79]
38         JOIN goede_node TO route OF tmagv
39         goede_node ← end_point OF outs[ 2] OF goede_node
40         JOIN goede_node TO route OF tmagv
41         END
42
43     goede_node←nodes[ 212+CEIL((tmx-1)/2)*30]
44     JOIN goede_node TO route OF tmagv
45     GOTO later
46     END
47
48     IF (tmx ≥7 & tmx≤12) & (CEIL(tmy/60)=1|CEIL(tmy/60)=3)
49
50     goede_node ← nodes[ CEIL(tmy/20)*30+200]
51     JOIN goede_node TO route OF tmagv
52
53         IF tmx<(tmy/10) @de agv moet terug
54         goede_node ← nodes[ 118-(CEIL((tmx + 0.5)/2)-1)*2]
55         JOIN goede_node TO route OF tmagv
56         END
57
58         IF tmx≥(tmy/10) @de agv moet heen
59
60         goede_node ← end_point OF outs[ 2] OF goede_node
61         JOIN goede_node TO route OF tmagv
62         goede_node ← nodes[ (CEIL((tmx+0.5)/2))*2+79]
```

Line 64 and 65 the rgv has to cross the track again

line 68 is the entrance of a portainerlocation

In line **73 to 112** if this condition counts it means the rgv has to travel back, to the left.
tmx/6 =1 this means the rgv has to go to berth 1 & tmy/60 =2 this indicates the rgv is located in a stack behind berth 2

line 77 this is the exit node out of the stack area

After is has taken the exit node it has to cross two lanes **line 80&81** and **line 83&84**

In **line 86 to 89** the exit node of the top main route is given

In line 91 to 95 the rgv has to cross 2 lanes again

In line 97 the entrance of the portainerlocation is given

In **line 104 to 106** a rgv has to go from a stack located behind berth 3 to a portainerlocation at berth 2 goto tweede ring pakken refers to **line 75**

In line **109 to 119** If this condition counts it means rgv has to go foreword to the right

This conditions means a rgv is located in a stack behind berth 1 and has to go to a portainer location behind berth 2
This is the same as line **75 to 99** therefore the rgv has to continue from tweede ring pakken **line 75**

This means a rgv is located in a stack behind berth 2 and has to go to a portainerlocation at berth here also the rgv has to continue from **line 75**

If the rgv number is even this means the rgv has to take the bottom route

If this condition counts It means the rgv is located in a stack behind the berth it has to go to.
There are two possibilities depending on the portainerlocation (the x of the rgv)

```
63          JOIN goede_node TO route OF tmagv
64          goede_node ← end_point OF outs[ 2] OF goede_node
65          JOIN goede_node TO route OF tmagv
66          END
67
68      goede_node←nodes[ 212+CEIL((tmx-1)/2)*30]
69      JOIN goede_node TO route OF tmagv
70      GOTO later
71      END
72
73      IF CEIL(tmx/6)<CEIL(tmy/60)
74          IF CEIL(tmx/6)=1 & CEIL(tmy/60)=2 @de tweede ring pakken
75          Tweede_ring_pakken:
76
77          goede_node ← nodes[ CEIL(tmy/20)*30+200]
78          JOIN goede_node TO route OF tmagv
79
80          goede_node ← end_point OF outs[ 2] OF goede_node
81          JOIN goede_node TO route OF tmagv       @oversteken
82
83          goede_node ← end_point OF outs[ 2] OF goede_node
84          JOIN goede_node TO route OF tmagv       @oversteken
85
86              IF tmx<(tmy/10) @de agv moet terug
87              goede_node ← nodes[ 78-(CEIL((CEIL(tmx+0.5)/2)-1))*2]
88              JOIN goede_node TO route OF tmagv
89              END
90
91          goede_node ← end_point OF outs[ 2] OF goede_node
92          JOIN goede_node TO route OF tmagv       @oversteken
93
94          goede_node ← end_point OF outs[ 2] OF goede_node
95          JOIN goede_node TO route OF tmagv       @oversteken
96
97          goede_node ← nodes[ 212+CEIL((tmx-1)/2)*30]
98          JOIN goede_node TO route OF tmagv
99
100         GOTO later
101
102         END
103
104         IF CEIL(tmx/6)=2 & CEIL(tmy/60)=3 @tweede ring pakken
105         GOTO Tweede_ring_pakken
106         END
107     END
108
109     IF CEIL(tmx/6)>CEIL(tmy/60)
110
111         IF CEIL(tmx/6)=2 & CEIL(tmy/60)=1 @tweede ring pakken
112         GOTO Tweede_ring_pakken
113         END
114
115         IF CEIL(tmx/6)=3 & CEIL(tmy/60)=2 @tweede ring pakken
116         GOTO Tweede_ring_pakken
117         END
118     END
119 END
120
121 IF (agvnumber OF tmagv)MODULO(2)=0  @AGV is even, dus onderlangs)
122     GOTO later
123     IF tmx=CEIL(tmy/10)                 @achter de portainer in het stack
124
```

If (tmx)modulo(2) =1 the portainer location is odd and the rgv has to go to the right (foreword)
For example a rgv is located in stack 44 it has to take exit 273 **(line 130)** than it has to take exit 506 **line 132**

If (tmx)modulo(2) =0 the portainer location is even and the rgv has to go to the left (back)
Line 137 is the same as **line 130**

Because the rgv has to go back it has to cross a lane **(line 139)**

This is the exit to the right portainerlocation

The rgv has to cross the lane again

This is the entrance of the portainerlocation **line 147**

In **line 157 to 183** The rgv is located in a stack behind berth 1 or 3 and has to go to berth 2
This is the same as **line 46 to 71** except here the rgv's take the bottom route instead of the top route

In line 185 the rgv is located in a stack behind berth 2 and has to go to berth 1
This is the same as **line 75** except this time the rgv takes the bottom route instead of the top route

```
125     @de crane heeft al een entrance node gegeven@
126     @dit is een andere dan bij de oneven agv's@
127
128         IF (tmx)MODULO(2)=1             @de agv moet heen
129
130         goede_node←nodes[ 213+30*CEIL(CEIL(((tmy/10)-1)/2))]
131         JOIN goede_node TO route OF tmagv
132         goede_node←nodes[ 500+ CEIL((CEIL(tmy/10)/2))*2]
133         JOIN goede_node TO route OF tmagv
134         END
135
136         IF (tmx)MODULO(2)=0                 @de agv moet terug
137         goede_node←nodes[ 213+CEIL((tmy/10-1)/2)*30]
138         JOIN goede_node TO route OF tmagv
139         goede_node ← end_point OF outs[ 2] OF goede_node
140         JOIN goede_node TO route OF tmagv       @oversteken
141         goede_node←nodes[ 539-(CEIL(CEIL(tmx))/2)*2]
142         JOIN goede_node TO route OF tmagv
143         goede_node ← end_point OF outs[ 2] OF goede_node
144         JOIN goede_node TO route OF tmagv       @oversteken
145         END
146
147     goede_node ← nodes[ 188+(CEIL(tmx/2))*30]
148     JOIN goede_node TO route OF tmagv
149
150     GOTO later
151
152     END
153
154
155     @IF CEIL(tmx/6)=CEIL(tmy/60) @stack zit in dezelfde berth
156
157     IF (tmx ≥7 & tmx≤12) & (CEIL(tmy/60)=1|CEIL(tmy/60)=3)
158
159         @AGV van 2 in berth op 2 gaat terug naar stack 1 of 3 via dichbij baan
160         @de crane heeft al een entrance node gegeven@
161         @dit is een andere dan bij de oneven agv's@
162
163         goede_node←nodes[ 213+CEIL(((tmy/10)-1)/2)*30]
164         JOIN goede_node TO route OF tmagv
165
166         IF tmx≥(tmy/10) @de agv moet heen
167         goede_node←nodes[ 500+ CEIL((CEIL(tmx)/2))*2]
168         JOIN goede_node TO route OF tmagv
169         END
170
171         IF tmx<(tmy/10)                 @de agv moet terug
172         goede_node ← end_point OF outs[ 2] OF goede_node
173         JOIN goede_node TO route OF tmagv       @oversteken
174         goede_node←nodes[ 539- CEIL(CEIL((tmx)/2))*2]
175         JOIN goede_node TO route OF tmagv
176         goede_node ← end_point OF outs[ 2] OF goede_node
177         JOIN goede_node TO route OF tmagv       @oversteken
178         END
179
180     goede_node ← nodes[ 188+(CEIL(tmx/2))*30]
181     JOIN goede_node TO route OF tmagv
182     GOTO Later
183     END
184
185     IF CEIL(tmx/6)=1 & CEIL(tmy/60)=2 @tweede ring van onderen
186     Tweede_ring_van_onderen:
```

The rgv is located in a stack behind berth 3 and has to go to berth 2 here is referred to **line 186**

The rgv is located in a stack behind berth 1 and has to go to berth 2 here is referred to **line 186**

The rgv Located in a stack behind berth 2 and has to go to berth 3 here is referred to **line 186**

```
187     @de crane heeft al een entrance node gegeven@
188     @dit is een andere dan bij de oneven agv's@
189
190     goede_node←nodes[ 213+CEIL(((tmy/10)-1)/2)*30]
191     JOIN goede_node TO route OF tmagv
192
193     goede_node ← end_point OF outs[ 2] OF goede_node
194     JOIN goede_node TO route OF tmagv       @oversteken
195
196     goede_node ← end_point OF outs[ 2] OF goede_node
197     JOIN goede_node TO route OF tmagv       @oversteken
198
199
200
201         IF tmx≥(tmy/10) @de agv moet heen
202         goede_node←nodes[ 540+(CEIL(tmx/2))*2]
203         JOIN goede_node TO route OF tmagv
204         END
205
206         IF tmx<(tmy/10)                 @de agv moet terug
207         goede_node ← end_point OF outs[ 2] OF goede_node
208         JOIN goede_node TO route OF tmagv       @oversteken
209
210         goede_node←nodes[ 579 - (CEIL(tmx/2))*2]
211         JOIN goede_node TO route OF tmagv
212
213         goede_node ← end_point OF outs[ 2] OF goede_node
214         JOIN goede_node TO route OF tmagv       @oversteken
215
216         END
217
218     goede_node ← end_point OF outs[ 2] OF goede_node
219     JOIN goede_node TO route OF tmagv       @oversteken
220
221     goede_node ← end_point OF outs[ 2] OF goede_node
222     JOIN goede_node TO route OF tmagv       @oversteken
223
224     goede_node ← nodes[ 188+(CEIL(tmx/2))*30]
225     JOIN goede_node TO route OF tmagv
226
227     GOTO later
228     END
229
230
231     IF CEIL(tmx/6)=2 & CEIL(tmy/60)=3
232     GOTO Tweede_ring_van_onderen
233     END
234
235     IF CEIL(tmx/6)=2 & CEIL(tmy/60)=1
236     GOTO Tweede_ring_van_onderen
237     END
238
239
240     IF CEIL(tmx/6)=3 & CEIL(tmy/60)=2
241     GOTO Tweede_ring_van_onderen
242     END
243 END
244 later:
245 RETURN
246
```

Agv_berth_stack is a macro that creates the route from a berth to the stack
The idea of this macro is the same as the macro agv_stack_berth off course the routes created are different

Line 9 to 14
Firstshuffle is the transition for a rgv from the loading of a vessel to shuffling the first shuffle container is retrieved from the stacks behind the portainerlocation the rgv was working
Easyload is an attribute of a rgv referring to the easy loading method of a vessel, this rgv retri
the load containers from the stacks behind the berth
In line 11 the node referred to is the entrance of a stack is joined to the route of the rgv

In **line 17 to 185** the difference is made between a odd and a even rgv
Odd rgv's take the top route at the terminal when they want to get somewhere at the terminal
Even rgv's take the bottom route when driving at the terminal
If the condition in **line 18** counts the rgv has to go from a portainerlocation to a stack located behind this portainerlocation (**Line 17 to 42)**

If the condition in **line 44** counts the rgv has to go from a portainerlocation at a berth to a stack located behind the same berth **Line 44 to 70**
Line 46 this condition states the rgv is at berth 2 and has to go to stack located behind berth 1 or 3

```
begin:
@
agvstack OF tmagv←FIRST stack IN stacklist WITH stacknumber = y OF tmagv

tmy←y OF tmagv
tmx←x OF tmagv


IF easyload OF tmagv = TRUE | firstshuffle OF tmagv = TRUE

    goede_node ← nodes[ 200-(tmy-1)MODULO(10)+ CEIL(tmy/20)*30]
    JOIN goede_node TO route OF tmagv
    GOTO later
END


IF (agvnumber OF tmagv)MODULO(2)=1  @ agv is oneven dus bovenlangs
    IF tmx=CEIL(tmy/10)                  @achter de portainer in het stack

    @de portainer heeft al een entrance node gegeven@

    goede_node ← nodes[ CEIL(tmx/2)*30+200]
    JOIN goede_node TO route OF tmagv

        IF (tmx)MODULO(2)=1              @de agv moet terug
        goede_node ← nodes[ 118-(CEIL(((CEIL(tmy/10)+0.5)/2)-1))*2]
        JOIN goede_node TO route OF tmagv
        END

        IF (tmx)MODULO(2)=0              @de agv moet heen
        goede_node ← end_point OF outs[ 2] OF goede_node
        JOIN goede_node TO route OF tmagv
        goede_node ← nodes[ (CEIL((CEIL(tmy/10)+0.5)/2))*2+79]
        JOIN goede_node TO route OF tmagv
        goede_node ← end_point OF outs[ 2] OF goede_node
        JOIN goede_node TO route OF tmagv
        END

    goede_node←nodes[ 200+tmy+10*FLOOR(tmy/20.1)+20*((tmy-1)MODULO(2))]
    JOIN goede_node TO route OF tmagv
    GOTO later
    END

    @IF CEIL(tmx/6)=CEIL(tmy/60) @stack zit in dezelfde berth

    IF (tmx ≥7 & tmx≤12) & (CEIL(tmy/60)=1|CEIL(tmy/60)=3)
        @AGV van 2 in berth op 2 gaat terug naar stack 1 of 3 via dichbij baan
        @de portainer heeft al een entrance node gegeven@

        goede_node ← nodes[ CEIL(tmx/2)*30+200]
        JOIN goede_node TO route OF tmagv

        IF tmx≥(tmy/10) @de agv moet terug
            goede_node ← nodes[ 118-(CEIL(((CEIL(tmy/10)+0.5)/2)-1))*2]
            JOIN goede_node TO route OF tmagv
        END

        IF tmx<(tmy/10) @de agv moet heen
            goede_node ← end_point OF outs[ 2] OF goede_node
            JOIN goede_node TO route OF tmagv
            goede_node ← nodes[ (CEIL((CEIL(tmy/10)+0.5)/2))*2+79]
            JOIN goede_node TO route OF tmagv
```

In line **72 to 116** the rgv is located at a berth and has to go to a stack this stack is located at the right side of the portainerlocation. If the condition in **Line 73** counts the rgv is at berth 1 and has to go to a stack located behind berth 2

Line 113 to 115 refers to a rgv at berth 2 that has to go to a stack located behind berth 3

If the condition in **line 118** counts the rgv is a berth and has to go to a stack at the left side of this portainerlocation
Line 120 to 121 refer to a rgv at berth 2 that has to go to a portainerlocation
tweede ring pakken in **line 121** refers to **line 74**

If this condition counts the rgv is located at berth 3 and has to go to a portainerlocation behind berth 2 **(line 124)**

```
63              goede_node ← end_point OF outs[ 2] OF goede_node
64              JOIN goede_node TO route OF tmagv
65          END
66
67          goede_node←nodes[ 200+tmy+10*FLOOR(tmy/20.1)+20*((tmy-1)MODULO(2))]
68          JOIN goede_node TO route OF tmagv
69          GOTO later
70      END
71
72      IF CEIL(tmx/6)<CEIL(tmy/60)
73          IF CEIL(tmx/6)=1 & CEIL(tmy/60)=2 @de tweede ring pakken
74          Tweede_ring_pakken:
75          @de portainer heeft al een entrance node gegeven@
76
77          goede_node ← nodes[ CEIL(tmx/2)*30+200]
78          JOIN goede_node TO route OF tmagv
79
80          goede_node ← end_point OF outs[ 2] OF goede_node
81          JOIN goede_node TO route OF tmagv       @oversteken
82
83          goede_node ← end_point OF outs[ 2] OF goede_node
84          JOIN goede_node TO route OF tmagv       @oversteken
85
86              IF tmx≥(tmy/10) @de agv moet terug
87              goede_node ← nodes[ 78-(CEIL(((CEIL(tmy/10)+0.5)/2)-1))*2]
88              JOIN goede_node TO route OF tmagv
89              END
90
91              IF tmx<(tmy/10) @de agv moet heen
92              goede_node ← end_point OF outs[ 2] OF goede_node
93              JOIN goede_node TO route OF tmagv
94              goede_node ← nodes[ (CEIL((CEIL(tmy/10)+0.5)/2))*2+39]
95              JOIN goede_node TO route OF tmagv
96              goede_node ← end_point OF outs[ 2] OF goede_node
97              JOIN goede_node TO route OF tmagv
98              END
99
100         goede_node ← end_point OF outs[ 2] OF goede_node
101         JOIN goede_node TO route OF tmagv       @oversteken
102
103         goede_node ← end_point OF outs[ 2] OF goede_node
104         JOIN goede_node TO route OF tmagv       @oversteken
105
106         goede_node ← nodes[ 200+tmy+10*FLOOR(tmy/20.1)+20*((tmy-1)MODULO(2))]
107         JOIN goede_node TO route OF tmagv
108
109         GOTO later
110
111         END
112
113         IF CEIL(tmx/6)=2 & CEIL(tmy/60)=3 @tweede ring pakken
114         GOTO Tweede_ring_pakken
115         END
116     END
117
118     IF CEIL(tmx/6)>CEIL(tmy/60)
119
120         IF CEIL(tmx/6)=2 & CEIL(tmy/60)=1 @tweede ring pakken
121         GOTO Tweede_ring_pakken
122         END
123
124                 IF CEIL(tmx/6)=3 & CEIL(tmy/60)=2 @tweede ring pakken
```

Line 152 tweede ring pakken refers to **line 74**

Line 130 to the end deals with the even rgv's that take the bottom route at the terminal

Line 132 to 159 deals with a rgv at a portainerlocation that has to go to a stack located behind that portainerlocation

Line 163 to 189 deals with a rgv at a portainerlocation behind berth 2 that has to go to a stack located behind berth 1 or berth 3

```
125         GOTO Tweede_ring_pakken
126         END
127     END
128 END
129
130 IF (agvnumber OF tmagv)MODULO(2)=0      @AGV is even, dus onderlangs)
131     GOTO later
132     IF tmx=CEIL(tmy/10)                 @achter de portainer in het stack
133
134     @de portainer heeft al een entrance node gegeven@
135     @dit is een andere dan bij de oneven agv's@
136     goede_node←FIRST node IN entrance OF tmagv
137     JOIN goede_node TO route OF tmagv
138     @dit is dezelfde als de entrance node
139
140         IF (tmx)MODULO(2)=1             @de agv moet heen
141         goede_node←nodes[ 500+ (CEIL((CEIL(tmy/10))/2))*2]
142         JOIN goede_node TO route OF tmagv
143         END
144
145         IF (tmx)MODULO(2)=0             @de agv moet terug
146         goede_node ← end_point OF outs[ 2] OF goede_node
147         JOIN goede_node TO route OF tmagv       @oversteken
148         goede_node←nodes[ 539-(CEIL((CEIL(tmy/10))/2))*2]
149         JOIN goede_node TO route OF tmagv
150         goede_node ← end_point OF outs[ 2] OF goede_node
151         JOIN goede_node TO route OF tmagv       @oversteken
152         END
153
154     goede_node ← nodes[ 200-(tmy-1)MODULO(10)+CEIL(tmy/20)*30]
155     JOIN goede_node TO route OF tmagv
156
157     GOTO later
158
159     END
160
161
162     @IF CEIL(tmx/6)=CEIL(tmy/60) @stack zit in dezelfde berth
163     IF (tmx ≥7 & tmx≤12) & (CEIL(tmy/60)=1|CEIL(tmy/60)=3)
164
165         @AGV van 2 in berth op 2 gaat terug naar stack 1 of 3 via dichbij baan
166         @de portainer heeft al een entrance node gegeven@
167         @dit is een andere dan bij de oneven agv's@
168         goede_node←FIRST node IN entrance OF tmagv
169         JOIN goede_node TO route OF tmagv
170         @dit is dezelfde als de entrance node
171
172         IF x≥(tmy/10) @de agv moet terug
173             goede_node ← end_point OF outs[ 2] OF goede_node
174             JOIN goede_node TO route OF tmagv       @oversteken
175             goede_node←nodes[ 539-(CEIL((CEIL(tmy/10))/2))*2]
176             JOIN goede_node TO route OF tmagv
177             goede_node ← end_point OF outs[ 2] OF goede_node
178             JOIN goede_node TO route OF tmagv       @oversteken
179         END
180
181         IF tmx<(tmy/10)                 @de agv moet heen
182             goede_node←nodes[ 500+ (CEIL((CEIL(tmy/10))/2))*2]
183             JOIN goede_node TO route OF tmagv
184         END
185
186         goede_node ← nodes[ 200-(tmy-1)MODULO(10)+ CEIL(tmy/20)*30]
```

Line 191 to 229 deals with a rgv at berth 1 that has to go to a stack located behind berth 2

line 234 deals with a rgv that is located at berth 2 and has to go to a stack located behind berth 3
goto tweede ring van onderen refers to line 196

line 238 deals with a rgv that is located at berth 2 and has to go to a stack located behind berth1
goto tweede ring van onderen refers to line 196

line 244 deals with a rgv that is located at berth 3 and has to go to a stack located behind berth2
goto tweede ring van onderen refers to line 196

```
187         JOIN goede_node TO route OF tmagv
188         GOTO Later
189     END
190
191     IF CEIL(tmx/6)=1 & CEIL(tmy/60)=2 @tweede ring van onderen
192
193     @de portainer heeft al een entrance node gegeven@
194     @dit is een andere dan bij de oneven agv's@
195
196     Tweede_ring_van_onderen:
197     goede_node←FIRST node IN entrance OF tmagv
198     JOIN goede_node TO route OF tmagv
199     @dit is dezelfde als de entrance node
200
201     goede_node ← end_point OF outs[ 2] OF goede_node
202     JOIN goede_node TO route OF tmagv       @oversteken
203
204     goede_node ← end_point OF outs[ 2] OF goede_node
205     JOIN goede_node TO route OF tmagv       @oversteken
206
207         IF tmx≥(tmy/10) @de agv moet terug
208         goede_node ← end_point OF outs[ 2] OF goede_node
209         JOIN goede_node TO route OF tmagv       @oversteken
210         goede_node←nodes[ 579-(CEIL((CEIL(tmy/10))/2))*2]
211         JOIN goede_node TO route OF tmagv
212         goede_node ← end_point OF outs[ 2] OF goede_node
213         JOIN goede_node TO route OF tmagv       @oversteken
214         END
215
216         IF tmx<(tmy/10)                 @de agv moet heen
217         goede_node←nodes[ 540+ (CEIL((CEIL(tmy/10))/2))*2]
218         JOIN goede_node TO route OF tmagv
219         END
220     goede_node ← end_point OF outs[ 2] OF goede_node
221     JOIN goede_node TO route OF tmagv       @oversteken
222
223     goede_node ← end_point OF outs[ 2] OF goede_node
224     JOIN goede_node TO route OF tmagv       @oversteken
225
226     goede_node ← nodes[ 200-(tmy-1)MODULO(10)+(CEIL(tmy/20))*30]
227     JOIN goede_node TO route OF tmagv
228
229     GOTO later
230     END
231
232
233
234     IF CEIL(tmx/6)=2 & CEIL(tmy/60)=3
235     GOTO Tweede_ring_van_onderen
236     END
237
238     IF CEIL(tmx/6)=2 & CEIL(tmy/60)=1
239     GOTO Tweede_ring_van_onderen
240     END
241
242
243
244     IF CEIL(tmx/6)=3 & CEIL(tmy/60)=2
245     GOTO Tweede_ring_van_onderen
246     END
247 END
248 later:
```

```
249 RETURN
250
```

agv_stack_stack is a macro that creates the route from a stack to another stack
This macro works the same as the macro's agv_berth_stack and avg_stack_berth

The odd and the even rgv's take different routes the odd rgv's take the top route
If the condition in **line 26** counts the rgv has to go from a stack to a stack located behind the same portainerlocation (**Line 25 to 36)**

In **line 40 to 69** the rgv is located in a stack behind berth 2 and has to go a stack located behind berth 1 or 3

```
 1
 2  goede_stack←FIRST stack IN stacklist WITH stacknumber = yy OF tmagv
 3 goede_section ←way_out_stack OF goede_stack @agvstack OF tmagv
 4 goede_node ←end_point OF goede_section
 5 JOIN goede_node TO entrance OF tmagv IF entrance OF tmagv IS EMPTY
 6
 7 tmyy←yy OF tmagv
 8 tmx ← x OF tmagv
 9 tmy ← y OF tmagv
10
11
12
13 IF easyload OF tmagv = TRUE
14
15     goede_node ← nodes[ (CEIL((tmx+0.5)/2))*30+181]
16     JOIN goede_node TO route OF tmagv
17
18     goede_node ← nodes[ 200-(tmy-1)MODULO(10)+CEIL((tmy/20))*30]
19     JOIN goede_node TO route OF tmagv
20
21     GOTO later
22 END
23
24
25 IF (agvnumber OF tmagv)MODULO(2)=1 |(AGVNUMBER OF TMAGV)MODULO(2)=0
26     IF CEIL(tmyy/10)=CEIL(tmy/10)       @de stacks liggen in hetzelfde cluster
27
28         goede_node ← nodes[ (CEIL((tmx+0.5)/2))*30+181]
29         JOIN goede_node TO route OF tmagv
30
31
32
33         goede_node←nodes[ 200+y+10*(FLOOR(tmy/20.1))+20*((tmy-1)MODULO(2))]
34         JOIN goede_node TO route OF tmagv
35         GOTO later
36     END
37
38
39
40     IF (tmx ≥7 & tmx≤12) @& (CEIL(tmy/60)=1|CEIL(tmy/60)=3)
41         @AGV van 2 in berth op 2 gaat terug naar stack 1 of 3 via dichbij baan
42
43         IF shuffleagv OF tmagv = FALSE | firstshuffle OF tmagv = TRUE
44             goede_node ← nodes[ (CEIL((tmx+0.5)/2))*30+181]
45             JOIN goede_node TO route OF tmagv
46             firstshuffle OF tmagv ← FALSE
47         END
48
49         goede_node ← nodes[ CEIL(tmyy/20)*30+200]
50         JOIN goede_node TO route OF tmagv
51
52         IF (tmyy/10)≥(tmy/10) @de agv moet terug
53             goede_node ← nodes[ 118-(CEIL(((CEIL(tmy/10)+0.5)/2)-1))*2]
54             JOIN goede_node TO route OF tmagv
55         END
56
57         IF (tmyy/10)<(tmy/10) @de agv moet heen
58             goede_node ← end_point OF outs[ 2] OF goede_node
59             JOIN goede_node TO route OF tmagv
60             goede_node ← nodes[ (CEIL((CEIL(tmy/10)+0.5)/2))*2+79]
61             JOIN goede_node TO route OF tmagv
62             goede_node ← end_point OF outs[ 2] OF goede_node
```

In line 74 to 121 the rgv is located in a stack behind berth 1 or 3

line 123 to the end deals with the even rgv's

```
63              JOIN goede_node TO route OF tmagv
64          END
65
66          goede_node←nodes[ 200+tmy+10*(FLOOR(tmy/20.1))+20*((tmy-1)MODULO(2))]
67          JOIN goede_node TO route OF tmagv
68          GOTO later
69      END
70
71
72      IF (tmx ≥13 | tmx≤6)
73
74          Tweede_ring_pakken:
75
76              IF shuffleagv OF tmagv = FALSE | firstshuffle OF tmagv = TRUE
77                  goede_node ← nodes[ (CEIL((tmx+0.5)/2))*30+181]
78                  JOIN goede_node TO route OF tmagv
79                  firstshuffle OF tmagv ← FALSE
80              END
81
82              goede_node ← nodes[ (CEIL(tmyy/20))*30+200]
83              JOIN goede_node TO route OF tmagv
84
85              goede_node ← end_point OF outs[ 2] OF goede_node
86              JOIN goede_node TO route OF tmagv       @oversteken
87
88              goede_node ← end_point OF outs[ 2] OF goede_node
89              JOIN goede_node TO route OF tmagv       @oversteken
90
91              IF (tmyy/10)≥(tmy/10) @de agv moet terug
92                  goede_node ← nodes[ 78-(CEIL(((CEIL(tmy/10)+0.5)/2)-1))*2]
93                  JOIN goede_node TO route OF tmagv
94              END
95
96              IF (tmyy/10)<(tmy/10) @de agv moet heen
97                  goede_node ← end_point OF outs[ 2] OF goede_node
98                  JOIN goede_node TO route OF tmagv
99                  goede_node ← nodes[ (CEIL((CEIL(tmy/10)+0.5)/2))*2+39]
100                 JOIN goede_node TO route OF tmagv
101                 goede_node ← end_point OF outs[ 2] OF goede_node
102                 JOIN goede_node TO route OF tmagv
103             END
104
105             goede_node ← end_point OF outs[ 2] OF goede_node
106             JOIN goede_node TO route OF tmagv       @oversteken
107
108             goede_node ← end_point OF outs[ 2] OF goede_node
109             JOIN goede_node TO route OF tmagv       @oversteken
110
111             goede_node ← nodes[ 200+tmy+10*(FLOOR(tmy/20.1))+20*((tmy-1)MODULO(2))]
112             JOIN goede_node TO route OF tmagv
113
114             GOTO later
115
116         END
117
118
119 END
120
121 IF (agvnumber OF tmagv)MODULO(2)=0      @AGV is even, dus onderlangs)
122
123     @@@@@@@@@@@@@@
124     @Zelfde stack@
```

line 125 to 139 concerns a rgv going from a stack behind a portainerlocation to a stack behind the same portainerlocation

Line 141 to 172concerns a rgv going from a stack behind a berth to a stack behind the same berth

Line 174 to 216 concerns a rgv going from a stack behind berth 1 or 3 to a stack behind 2

```
125     @@@@@@@@@@@@@@
126     IF CEIL(tmyy/10)=CEIL(tmy/10)   @achter de portainer in het stack
127                                 @dit is een andere dan bij de oneven agv's@
128
129         goede_node ← nodes[ (CEIL((tmyy-10)/20)+1)*30+183]
130         JOIN goede_node TO route OF tmagv
131
132         goede_node ← nodes[ 200-(tmy-1)MODULO(10)+(CEIL(tmy/20))*30]
133         JOIN goede_node TO route OF tmagv
134
135         GOTO later
136
137     END
138
139     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
140     @stack zit in dezelfde berth@
141     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
142     @IF CEIL(tmyy/60)=CEIL(tmy/60)
143     IF (tmx ≥7 & tmx≤12) @& (CEIL(tmy/60)=1|CEIL(tmy/60)=3)
144
145         @AGV van 2 in berth op 2 gaat terug naar stack 1 of 3 via dichbij baan
146         @de crane heeft al een entrance node gegeven@
147         @dit is een andere dan bij de oneven agv's@
148
149         goede_node ← nodes[ (CEIL((tmyy-10)/20)+1)*30+183]
150         JOIN goede_node TO route OF tmagv
151
152
153         IF (tmyy/10)≥(tmy/10) @de agv moet terug
154             goede_node ← end_point OF outs[ 2] OF goede_node
155             JOIN goede_node TO route OF tmagv       @oversteken
156             goede_node←nodes[ 539-(CEIL((tmy/10)/2))*2]
157             JOIN goede_node TO route OF tmagv
158             goede_node ← end_point OF outs[ 2] OF goede_node
159             JOIN goede_node TO route OF tmagv       @oversteken
160         END
161
162         IF (tmyy/10)<(tmy/10)               @de agv moet heen
163             goede_node←nodes[ 500+ (CEIL((tmy/10)/2))*2]
164             JOIN goede_node TO route OF tmagv
165         END
166
167         goede_node ← nodes[ 200-(tmy-1)MODULO(10)+(CEIL(tmy/20))*30]
168         JOIN goede_node TO route OF tmagv
169         GOTO later
170     END
171
172     @@@@@@@@@@@@@@@@@@@@
173     @Aangrenzende berth@
174     @@@@@@@@@@@@@@@@@@@@
175     @IF CEIL(tmyy/60)=2 & CEIL(tmy/60)=1 @tweede ring van onderen
176     @de crane heeft al een entrance node gegeven@
177     @dit is een andere dan bij de oneven agv's@
178
179     IF (tmx ≥13 | tmx≤6)
180         Tweede_ring_van_onderen:
181
182         goede_node ← nodes[ (CEIL((tmyy-10)/20)+1)*30+183]
183         JOIN goede_node TO route OF tmagv
184
185         goede_node ← end_point OF outs[ 2] OF goede_node
186         JOIN goede_node TO route OF tmagv       @oversteken
```

```
187
188         goede_node ← end_point OF outs[ 2] OF goede_node
189         JOIN goede_node TO route OF tmagv       @oversteken
190
191         IF (tmyy/10)≥(tmy/10) @de agv moet terug
192             goede_node ← end_point OF outs[ 2] OF goede_node
193             JOIN goede_node TO route OF tmagv       @oversteken
194             goede_node←nodes[ 579-(CEIL(CEIL((tmy/10))/2))*2]
195             JOIN goede_node TO route OF tmagv
196             goede_node ← end_point OF outs[ 2] OF goede_node
197             JOIN goede_node TO route OF tmagv       @oversteken
198         END
199
200         IF (tmyy/10)<(tmy/10)              @de agv moet heen
201             goede_node←nodes[ 540+ (CEIL(CEIL((tmy/10))/2))*2]
202             JOIN goede_node TO route OF tmagv
203         END
204         goede_node ← end_point OF outs[ 2] OF goede_node
205         JOIN goede_node TO route OF tmagv       @oversteken
206
207         goede_node ← end_point OF outs[ 2] OF goede_node
208         JOIN goede_node TO route OF tmagv       @oversteken
209
210         goede_node ← nodes[ 200-(tmy-1)MODULO(10)+(CEIL(tmy/20))*30]
211         JOIN goede_node TO route OF tmagv
212
213         GOTO later
214     END
215
216     @@@@@@@@@@@@@@
217     @Berth tussen@
218     @@@@@@@@@@@@@@
219
220 END
221
222 later:
223 @hier staat iets nieuws
224 yyy OF tmagv←y OF tmagv
225 xxx OF tmagv←FLOOR((yyy OF tmagv-1)/20)
226 zzz OF tmagv←FLOOR((yyy OF tmagv-1)/20)
227 yyy OF tmagv←(y OF tmagv)-(zzz OF tmagv)*20
228
229 IF (yyy OF tmagv)MODULO(2)=0
230     yyy OF tmagv ← yyy OF tmagv -1
231 END
232
233 yyy OF tmagv ← yyy OF tmagv + 20*(xxx OF tmagv)
234 agvcrane ← FIRST crane IN cranelist  WITH cranenumber = yyy OF tmagv
235
236
237 JOIN tmagv TO assignedqueue OF agvcrane
238
239 RETURN
240
241
```

create_stack is the macro that creates the stacks this macro is called upon by the main

way_in_stack is a attribute of the stack referring to the way in

n_in is a reference to a node, the entrance of the stack

st_x is the actual x coordinate of the stack on the terminal
st_y is the actual y coordinate of the stack on the terminal

```
 1
 2 THIS stack ← NEW stack CALLED char
 3 stacknumber OF THIS stack←i
 4 IF (i)MODULO(2)=0
 5  stackcount OF THIS stack ← 0
 6 END
 7 wantasqueue←NEW SET
 8 way_in_stack ← NEW section
 9 way_out_stack ← NEW section
10 stack_row ← NEW SET WITH HISTORY
11 n_in ← READ FROM hconfig
12 n_out ← READ FROM hconfig
13 THIS NODE ← nodes[ n_in]
14 outs[ 2] ← way_in_stack
15 side_track[ 1] ← stack_crane
16 st_x ← READ FROM hconfig
17 st_y ← READ FROM hconfig
18 end_point OF way_in_stack ← THIS stack
19 s_length OF way_in_stack ← distance_to(st_x,st_y)
20 THIS node ← nodes[ n_out]
21 end_point OF way_out_stack ← THIS node
22 s_length OF way_out_stack ← distance_to(st_x,st_y)
23 JOIN THIS stack TO stacklist
24 i← i+1
25 RETURN
26
27
```

super_shortcut is a macro designed for an exception at the terminal. On some points at the terminal there are more shortcuts that can be taken this extra shortcut is than called a super shortcut

```
 1 @ The trolley passes a section being a short cut of the main route. @
 2 @ Trolleys can be delayed in this section, So, SEC_ROW is needed.   @
 3 @-------------------------------------------------------------------@
 4
 5 claimed OF cur_node ← claimed OF cur_node - 1
 6 cur_sec ← outs[ 3]  OF cur_node
 7
 8 WORK distance/node_speed
 9
10 WORK s_length OF cur_sec/low_speed
11 cur_node ← end_point OF cur_sec
12
13 REMOVE FIRST node IN superroute OF THIS agv  FROM superroute OF THIS agv
14
15 WAIT WHILE claimed OF cur_node >0
16 RETURN
```

unloadyardallocation is a macro that creates the stacks for the rgv's to deliver the unloaded containers

tmtotalim is the total of containers left on the vessel that have to be unloaded
tmportloc is the portlocation the rgv is working for
im40 of a vessel is the number of 40 feet import containers
tmres40im is the percentage 40 feet importcontainers of all the import containers
tmrestotim is the percentage import containers still to be unloaded of the whole

In **line 15 to 43** first the different kind of container, 40 ft or 20 ft is chosen. As a result of the kind of container the stack is chosen
If w1 <tmres40in, the rgv gets a 40 ft container **Line 15**

In line 16 to 18 the Im40 container = 0 this means the

im40 of this vessel becomes im40 -1, one 40 feet container is deducted this is a counter

If the condition **in line 30** counts the rgv will receive a 20 ft container

Line 44 to 52 concern the shuffling of the containers

```
 1 begin:
 2
 3 tmx ← x OF tmagv
 4
 5 tmtotalim ← (im40 OF tmship + im20 OF tmship)
 6 tmportloc ← FIRST portloc IN portlocations WITH portlocnumber = tmx
 7
 8 tmres40im ← (im40 OF tmship)/tmtotalim
 9 tmrestotim ← (im40 OF tmship + im20 OF tmship)/tmtotalim
10
11
12 w1 ← SAMPLE FROM unif
13 w2 ← SAMPLE FROM unif
14
15 IF  w1 < tmres40im
16     IF im40 OF tmship = 0 & tmtotalim > 0
17         GOTO begin
18     END
19     im40 OF tmship ← im40 OF tmship - 1
20
21     y OF tmagv ← 10*(tmx-1) + →
22     (tmx)MODULO(2)*((teller40im OF tmportloc)MODULO(6)+6)+ →
23     ((tmx)-1)MODULO(2)*((teller40im OF tmportloc)MODULO(6)+5)
24
25     teller40im OF tmportloc ← teller40im OF tmportloc+ 2
26     cont20ft OF tmagv ← FALSE
27 END
28
29
30 IF  w1 ≥ tmres40im & w1 < tmrestotim
31     IF im20 OF tmship = 0 & tmtotalim > 0
32         GOTO begin
33     END
34     im20 OF tmship ← im20 OF tmship - 1
35
36     y OF tmagv ← 10*(tmx - 1) + →
37     ((tmx)-1)MODULO(2)*((teller20im OF tmportloc)MODULO(4)+1)+ →
38     (tmx)MODULO(2)*((teller20im OF tmportloc)MODULO(4)+2)
39
40     teller20im OF tmportloc ← teller20im OF tmportloc+ 2
41     cont20ft OF tmagv ← TRUE
42 END
43
44 @--A shuffle destination berthnumber is given--@
45
46 w2 ← FLOOR((CEIL(w2*2))*0.5+(CEIL(w2*2)))    @w2 = 1 of 3
47 IF agvberthnb OF tmagv = 2
48     sbnr OF tmagv ← w2
49 END
50 IF agvberthnb OF tmagv = 1 |agvberthnb OF tmagv = 3
51     sbnr OF tmagv ← 2
52 END
53
54
55 RETURN
56
```

Loadyardallocation is a macro that creates the stacks where the rgv's have to get their containers

tmtotalex is the total of containers to be loaded unto the vessel - total already loaded

tex is the total of containers to be loaded unto the vessel

The odd rgv's take the top route of the terminal **line 33 to 93**

If this condition counts the rgv has to get a forty foot container **line 38**

In line 49 to 56 the actual stack the is chosen

If this condition counts the rgv has to get a twenty foot container **line 61**

```
begin:

yy OF tmagv ← y OF tmagv


tmx ← x OF tmagv
tmy ← y OF tmagv
tmyy← yy OF tmagv


tmportloc ← FIRST portloc IN portlocations WITH portlocnumber = tmx

IF tmtotalex = 0
    RETURN
END
IF tex OF tmship = 0
    RETURN
END
tmres40ex ← (ex40 OF tmship)/tmtotalex
tmrestotex ← (ex40 OF tmship + ex20 OF tmship)/tmtotalex

w1 ← SAMPLE FROM unif
w2 ← SAMPLE FROM unif
w3 ← SAMPLE FROM unif
w4 ← SAMPLE FROM unif
w5 ← SAMPLE FROM unif

IF inplacecounter[ agvberthnb OF tmagv]>0
    GOTO easyloading
END


@@@
```

In line 72 to 79 the actual stack is chosen

line 95 to 152 is about the even rgv's which take the bottom route

Line 100 if this condition counts the rgv has to get a 40 foot container

In line 110 to 117 the stack is chosen to get the 40 ft container

Line 122 if this condition counts a twenty foot container has to be retrieved from the stack

```
 63              GOTO begin
 64          END
 65          ex20 OF tmship ← ex20 OF tmship - 1
 66
 67          w2 ← CEIL(w2*2)
 68          w3 ← CEIL(w3*6)+6
 69          w4 ← CEIL(w4*2)
 70          w5 ← CEIL(w5*4)+2
 71
 72          @de bovenste is om de ptloc te bep@
 73          tmy ← ((CEIL(tmx/6))MODULO(2))*w3 + →
 74          ((CEIL(tmx/6)+1)MODULO(2))*((w5)+((w2-1)*10))
 75          w2 ← SAMPLE FROM unif
 76          w2 ← CEIL(w2*2)
 77          @de onderste is om het stack te bep@
 78          y OF tmagv ← ((tmy)-1)*10 + →
 79          (FLOOR((w4*0.5)*(((tmy)+1)MODULO(2)+3)))
 80
 81          cont20ft OF tmagv ← TRUE
 82      END
 83
 84      IF easyload OF tmagv =TRUE
 85          uneasy OF tmagv ← TRUE
 86          easyload OF tmagv ← FALSE
 87      END
 88
 89      easyload OF tmagv ← FALSE
 90
 91
 92      GOTO provided
 93  END
 94
 95  @@@@
```

In line 131 to 152 the stack is chosen to get the tweny foot container

easy loading concerns the loading where the containers are already in the stacks behind the right berth, this is only possible with shuffling

line 159 to 167 if this condition counts a 40 ft container has to be retrieved

line 168 to 176 if this condition counts a 20 ft container has to be retrieved

inplacecounter is the counter that counts the containers that are inplace behind that specific t

```
125         END
126         ex20 OF tmship ← ex20 OF tmship - 1
127         w2 ← CEIL(w2*2)
128         w3 ← CEIL(w3*6)+6
129         w5 ← CEIL(w5*4)+2
130
131         @de bovenste is om de ptloc te bep@
132         tmy ← ((CEIL(tmx/6))MODULO(2))*w3 + →
133         ((CEIL(tmx/6)+1)MODULO(2))*((w5)+((w2-1)*10))
134         w2 ← SAMPLE FROM unif
135         w2 ← CEIL(w2*2)
136         @de onderste is om het stack te bep@
137         y OF tmagv ← ((tmy)-1)*10 + →
138         (FLOOR((w2*0.5)*((tmy)MODULO(2)+3)))
139
140         cont20ft OF tmagv ← TRUE
141     END
142
143     IF easyload OF tmagv =TRUE
144         uneasy OF tmagv ← TRUE
145         easyload OF tmagv ← FALSE
146     END
147     easyload OF tmagv ← FALSE
148
149
150     GOTO provided
151 END
152
153 easyloading:
154
155 @@@@
```

```
187 GOTO easyloaded
188
189
190
191 provided:
192
193 @-- calculate the number of containers left in the stange berth --@
194 @--                      for the home berth                      --@
195
196 sbnr OF tmagv ← CEIL((y OF tmagv)/60)
197 frombtob[ sbnr OF tmagv,agvberthnb OF tmagv]←frombtob[ sbnr OF tmagv,agvberthnb OF
198
199 easyloaded:
200
201 RETURN
202 WAIT
```

conversation is the macro that creates the control box
control box is the box that appears when starting the run this enables the user to check the stackingcranes and some nodes after the run

```
@ ask user for configuration data @
@--------------------------------@

control ← NEW WINDOW CALLED "Control"
SPECIFY control ORIGIN(10,120) UNITS(5,5)
b_ok ← NEW WOB
JOIN b_ok TO control
n ← _wob_button(b_ok,0,0,10,5,14,1,100,"OK")

@ shortcut_node @
prm_mn ← NEW WOB
str_mn ← NEW WOB
n ← _wob_string(str_mn,0,20,10,0,10," 508",7)
n ← _wob_prompt(prm_mn,10,20,1,"shortcut1")
JOIN prm_mn TO control
JOIN str_mn TO control

prm_maxn ← NEW WOB
str_maxn ← NEW WOB
n ← _wob_string(str_maxn,0,14,10,0,10," 260",7)
n ← _wob_prompt(prm_maxn,10,14,1,"shortcut2")
JOIN prm_maxn TO control
JOIN str_maxn TO control

prm_mt ← NEW WOB
str_mt ← NEW WOB
n ← _wob_string(str_mt,25,20,10,0,10," 453",7)
n ← _wob_prompt(prm_mt,35,20,1,"shortcut3")
JOIN prm_mt TO control
JOIN str_mt TO control

prm_m ← NEW WOB
str_m ← NEW WOB
n ← _wob_string(str_m,25,14,10,0,10," 512",7)
n ← _wob_prompt(prm_m,35,14,1,"shortcut4")
JOIN prm_m TO control
JOIN str_m TO control



@ cranes @

prm_01 ← NEW WOB
str_01 ← NEW WOB
n ← _wob_string(str_01,0,8,10,3,13," 21",7)
n ← _wob_prompt(prm_01,10,8,1,"stack1")
JOIN prm_01 TO control
JOIN str_01 TO control

prm_02 ← NEW WOB
str_02 ← NEW WOB
n ← _wob_string(str_02,25,8,10,0,13," 22",7)
n ← _wob_prompt(prm_02,35,8,1,"stack2")
JOIN prm_02 TO control
JOIN str_02 TO control

prm_03 ← NEW WOB
str_03 ← NEW WOB
n ← _wob_string(str_03,50,8,10,3,13," 23",7)
n ← _wob_prompt(prm_03,60,8,1,"stack3")
JOIN prm_03 TO control
JOIN str_03 TO control
```

```
 63
 64 prm_04 ← NEW WOB
 65 str_04 ← NEW WOB
 66 n ← _wob_string(str_04,75,8,10,0,13," 24",7)
 67 n ← _wob_prompt(prm_04,85,8,1,"stack4")
 68 JOIN prm_04 TO control
 69 JOIN str_04 TO control
 70
 71 prm_05 ← NEW WOB
 72 str_05 ← NEW WOB
 73 n ← _wob_string(str_05,0,-8,10,3,13," 25",7)
 74 n ← _wob_prompt(prm_05,10,-8,1,"stack5")
 75 JOIN prm_05 TO control
 76 JOIN str_05 TO control
 77
 78 prm_06 ← NEW WOB
 79 str_06 ← NEW WOB
 80 n ← _wob_string(str_06,25,-8,10,0,13," 26",7)
 81 n ← _wob_prompt(prm_06,35,-8,1,"stack6")
 82 JOIN prm_06 TO control
 83 JOIN str_06 TO control
 84
 85 prm_07 ← NEW WOB
 86 str_07 ← NEW WOB
 87 n ← _wob_string(str_07,50,-8,10,3,13," 27",7)
 88 n ← _wob_prompt(prm_07,60,-8,1,"stack7")
 89 JOIN prm_07 TO control
 90 JOIN str_07 TO control
 91
 92 prm_08 ← NEW WOB
 93 str_08 ← NEW WOB
 94 n ← _wob_string(str_08,75,-8,10,0,13," 28",7)
 95 n ← _wob_prompt(prm_08,85,-8,1,"stack8")
 96 JOIN prm_08 TO control
 97 JOIN str_08 TO control
 98
 99 prm_09 ← NEW WOB
100 str_09 ← NEW WOB
101 n ← _wob_string(str_09,0,-16,10,3,13," 29",7)
102 n ← _wob_prompt(prm_09,10,-16,1,"stack9")
103 JOIN prm_09 TO control
104 JOIN str_09 TO control
105
106 prm_10 ← NEW WOB
107 str_10 ← NEW WOB
108 n ← _wob_string(str_10,25,-16,10,0,13," 30",7)
109 n ← _wob_prompt(prm_10,35,-16,1,"stack10")
110 JOIN prm_10 TO control
111 JOIN str_10 TO control
112
113 SHOW control AT(20,20) SIZE(500,340)
114
115
116
117 rep_cntr:
118 n ← _window_response(control)
119 n ← _wob_string_value(str_mn,<char9>)
120 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
121 mn ← char9
122
123 n ← _wob_string_value(str_maxn,<char9>)
124 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
```

```
125 mnb ← char9
126
127 n ← _wob_string_value(str_01,<char9>)
128 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
129 c1 ← char9
130
131 n ← _wob_string_value(str_02,<char9>)
132 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
133 c2 ← char9
134
135 n ← _wob_string_value(str_03,<char9>)
136 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
137 c3 ← char9
138
139 n ← _wob_string_value(str_04,<char9>)
140 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
141 c4 ← char9
142
143 n ← _wob_string_value(str_05,<char9>)
144 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
145 c5 ← char9
146
147 n ← _wob_string_value(str_06,<char9>)
148 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
149 c6 ← char9
150
151 n ← _wob_string_value(str_07,<char9>)
152 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
153 c7 ← char9
154
155 n ← _wob_string_value(str_08,<char9>)
156 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
157 c8 ← char9
158
159 n ← _wob_string_value(str_09,<char9>)
160 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
161 c9 ← char9
162
163 n ← _wob_string_value(str_10,<char9>)
164 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
165 c10 ← char9
166
167 n ← _wob_string_value(str_m,<char9>)
168 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
169 nm ← char9
170
171 n ← _wob_string_value(str_mt,<char9>)
172 REPEAT FROM rep_cntr IF numeric_control(<char9>) = FALSE
173 bmn ← char9
174 RETURN
```

Preclaimnodes is a macro called upon by the agv responsible for claiming nodes when more than one node has to be claimed in order to enter the main track
line 1 to 3 a rgv wants to enter a main track but it has to wait while the entrance node is claimed

Line 8 to 37 as many nodes will be claimed as needed for the rgv to enter the main track

In line 10 to 15 there are some exceptions, in the case of these nodes (line 10) only one node has to be claimed because they lie at the end of the main track (node 231,261,291 etc)

line 22 to 28 deals with an exception at the end of a main track

claimset is a set of nodes that have to be claimed if the length of the sections between the nodes is 30 it is possible for the rgv to enter the main track else another node has to be claimed in order to extend the claimedlength

claimedlength is the length the rgv needs to enter the main track

In line 42 to 46 the set is emptied and the claimed nodes are unclaimed

The rgv returns to the agvmodule where it came from **line 48** and enters the main track

The rgv has to wait while claimed of claim_node is not 0
goto again in **line 8** from there another node will be claimed until the claimedlength is >=30 meter

```
 1 IF claimed OF entrance_node ≠ 0
 2     WAIT WHILE claimed OF entrance_node ≠ 0
 3 END
 4
 5 claimedlength OF THIS agv ← 0
 6 b ← node_number OF entrance_node
 7
 8 again:
 9
10 IF b=231|b=261|b=291|b=321|b=351|b=381|b=411|b=441|b=471
11     claim_node ← FIRST node IN node_list WITH node_number = b
12     nodeagvnb OF claim_node ← agvnumber OF THIS agv
13     claimed OF claim_node ← claimed OF claim_node + 1
14     ACTIVATE claim_node FROM begin IN nodemod
15     GOTO empty_the_set
16 END
17
18 b←b-1
19 claim_node ← FIRST node IN node_list WITH node_number = b
20 claim_sec ← outs[ 1]  OF claim_node
21
22 IF claim_sec IS NONE
23     claim_node ← FIRST node IN node_list WITH node_number = b+1
24     nodeagvnb OF claim_node ← agvnumber OF THIS agv
25     claimed OF claim_node ← claimed OF claim_node + 1       @Claimed wordt 2
26     ACTIVATE claim_node FROM begin IN nodemod
27     GOTO empty_the_set
28 END
29
30 IF claimed OF claim_node = 0
31     claimed OF claim_node ← claimed OF claim_node + 1
32     claimedlength OF THIS agv ← s_length OF claim_sec + claimedlength OF THIS agv
33     JOIN claim_node TO claimset
34
35     IF claimedlength < 30
36
37         GOTO again
38     END
39
40     empty_the_set:
41     Le←LENGTH OF claimset
42     FOR i ← 1 TO Le
43         claim_node ← FIRST node IN claimset
44         REMOVE FIRST node IN claimset FROM claimset
45         claimed OF claim_node  ←  claimed OF claim_node - 1
46     END
47
48 RETURN
49 END
50
51 WAIT WHILE claimed OF claim_node ≠ 0
52 GOTO again
```

agv_uneasy_b_s is a macro that creates a route from the berth to a stack
This macro is called upon by the trafficmaster in **line 282 in the trafficmastermodule**, this case is the transition between loading a vessel and shuffling and only counts for a even rgv

Line 16 and 17 the exit of the portainerlocation becomes a point for the route of the rgv this means the rgv has to leave the main track the moment it enters the main track behind the portainer

If this condition counts the rgv is at berth 2 and has to go to a stack behind berth 1 or the rgv is at berth 3 in both cases the rgv has to go back **line 19 to 50**

If this condition counts the rgv os at berth 1 and has to go to a stack located behind berth 2 or the rgv is at berth 2 and has to go to a stack located behind berth 3 **line 52 to 88**

```
 1 begin:
 2
 3 tmy←y OF tmagv
 4 tmx←x OF tmagv
 5 tmyy←yy OF tmagv
 6
 7 agvstack←FIRST stack IN stacklist WITH stacknumber = tmy
 8
 9
10 IF shuffleagv OF tmagv = TRUE
11     GOTO uneasy_first_shuffle
12 END
13
14 uneasy OF tmagv ← FALSE
15
16 cur_node ← nodes[ CEIL(tmx/2)*30+200]
17 JOIN cur_node TO route OF tmagv
18
19     IF CEIL(tmx/6)=3|(CEIL(tmx/6)=2&CEIL(tmy/60)=1)@de agv gaat terug
20
21         cur_node ← nodes[ 118-(CEIL((((tmx-1)+0.5)/2)-1))*2]
22         JOIN cur_node TO route OF tmagv
23
24         cur_node←nodes[ 213+CEIL((tmx-2)/2)*30]
25         JOIN cur_node TO route OF tmagv
26
27         cur_node ← end_point OF outs[ 2] OF cur_node
28         JOIN cur_node TO route OF tmagv     @oversteken
29
30         cur_node ← end_point OF outs[ 2] OF cur_node
31         JOIN cur_node TO route OF tmagv     @oversteken
32
33         cur_node ← end_point OF outs[ 2] OF cur_node
34         JOIN cur_node TO route OF tmagv     @oversteken
35
36         cur_node←nodes[ 579-(CEIL((CEIL(tmy/10))/2))*2]
37         JOIN cur_node TO route OF tmagv
38
39         cur_node ← end_point OF outs[ 2] OF cur_node
40         JOIN cur_node TO route OF tmagv
41
42         cur_node ← end_point OF outs[ 2] OF cur_node
43         JOIN cur_node TO route OF tmagv     @oversteken
44
45         cur_node ← end_point OF outs[ 2] OF cur_node
46         JOIN cur_node TO route OF tmagv     @oversteken
47
48         cur_node ← nodes[ 200-(tmy-1)MODULO(10)+(CEIL(tmy/20))*30]
49         JOIN cur_node TO route OF tmagv
50     END
51
52     IF CEIL(tmx/6)=1|(CEIL(tmx/6)=2&CEIL(tmy/60)=3) @de agv gaat heen
53
54         cur_node ← end_point OF outs[ 2] OF cur_node
55         JOIN cur_node TO route OF tmagv
56
57         cur_node ← end_point OF outs[ 2] OF cur_node
58
59         cur_node ← end_point OF outs[ 1] OF cur_node
60         JOIN cur_node TO route OF tmagv
61
62         cur_node ← end_point OF outs[ 2] OF cur_node
```

uneasy first shuffle creates a route for the rgv to the first shuffle location **line 91 to 123**

```
63          JOIN cur_node TO route OF tmagv
64
65          cur_node←nodes[ 213+CEIL((tmx)/2)*30]
66          JOIN cur_node TO route OF tmagv
67
68          cur_node ← end_point OF outs[ 2] OF cur_node
69          JOIN cur_node TO route OF tmagv
70
71          cur_node ← end_point OF outs[ 2] OF cur_node
72          JOIN cur_node TO route OF tmagv
73
74          cur_node←nodes[ 540+(CEIL((CEIL(tmy/10))/2))*2]
75          JOIN cur_node TO route OF tmagv
76
77          cur_node ← end_point OF outs[ 2] OF cur_node
78          JOIN cur_node TO route OF tmagv     @oversteken
79
80          cur_node ← end_point OF outs[ 2] OF cur_node
81          JOIN cur_node TO route OF tmagv     @oversteken
82
83          cur_node ← nodes[ 200-(tmy-1)MODULO(10)+(CEIL(tmy/20))*30]
84          JOIN cur_node TO route OF tmagv
85
86      END
87
88
89  RETURN
90
91  uneasy_first_shuffle:
92
93  IF (tmx)MODULO(2)= 0
94
95      JOIN cur_node TO route OF tmagv
96
97      cur_node ← end_point OF outs[ 2] OF cur_node
98      JOIN cur_node TO route OF tmagv
99
100     cur_node ← end_point OF outs[ 2] OF cur_node
101
102     cur_node ← end_point OF outs[ 1] OF cur_node
103     JOIN cur_node TO route OF tmagv
104
105     cur_node ← end_point OF outs[ 2] OF cur_node
106     JOIN cur_node TO route OF tmagv
107
108     cur_node ← nodes[ 200-(tmy-1)MODULO(10)+(CEIL(tmy/20))*30]
109     JOIN cur_node TO route OF tmagv
110
111 END
112
113 IF (tmx)MODULO(2)= 1
114
115     JOIN cur_node TO route OF tmagv
116
117     cur_node ← nodes[ 500+(tmx)+1]
118     JOIN cur_node TO route OF tmagv
119
120     cur_node ← nodes[ 200-(tmy-1)MODULO(10)+(CEIL(tmy/20))*30]
121     JOIN cur_node TO route OF tmagv
122
123 END
124
```

```
125 RETURN
```

yardshuffleallocation is the macro that creates the locations for the rgv to retrieve containers

In line 16 to 69

Line 20 the second condition says the rgv is at his portainerlocation or in the stack behind its own portainerlocation
firstshuffle is an attribute of the rgv and indicates this rgv is shuffling for the first time

Depending of the kind of container a stack is chosen where to get the first shuffle container **line 26 to 36** deals with the 20 ft containers

Line 37 to 47 deals with the 40 ft containers

sbnr is the stack berth number and identifies behind which berth the stacks are located where the rgv has to go. If the rgv is at berth 1 or 3 (condition in **line 52**) the rgv has to go berth 2

If the rgv is at berth 2 it has to go to berth 1 or 3 **line 55**

```
 1 begin:
 2
 3 shuffleagv OF tmagv ← TRUE
 4
 5 tmy ← y OF tmagv
 6 tmx ← x OF tmagv
 7 tmyy ← yy OF tmagv
 8
 9
10 w1 ← SAMPLE FROM unif
11 w2 ← SAMPLE FROM unif
12 w3 ← SAMPLE FROM unif
13 w4 ← SAMPLE FROM unif
14 w5 ← SAMPLE FROM unif
15 w6 ← SAMPLE FROM unif
16 @@@@
```

line 69 to 90 is about collecting a container under the crane at its own stack not for the first time

Line 92 to 189 is about the delivering of a container from one stack to another starting at a stack behind the rgv's own portainerlocation

If the conditions in **line 96** count the rgv is loaded in a stack behind its own portainerlocation

if sbnr=0 it indicates the rgv is back in is own stack ready to go to another stack **Line 98 to 138**

```
63 @@@@
```

Line 140 to 182 is almost the same as line **98 to 138** except for the fact that it is the first time for the rgv

```
125         IF cont20ft OF tmagv = FALSE
126             IF (agvnumber OF tmagv)MODULO(2)=1  @bovenlangs
127                 y OF tmagv ← ((tmy)-1)*10 + →
128                 ((2*w4)+(((tmy)-1)MODULO(2))+3)
129             END
130
131             IF (agvnumber OF tmagv)MODULO(2)=0  @onderlangs
132                 y OF tmagv ← ((tmy)-1)*10 + →
133                 ((2*w4)+((tmy)MODULO(2))+3)
134             END
135         END
136
137         GOTO almostready
138     END
139
140     IF sbnr OF tmagv ≠ 0
141         yy OF tmagv ← y OF tmagv
142
143         w1 ← CEIL(w1*2)
144         w2 ← CEIL(w2*2)
145         w3 ← CEIL(w3*4)+6
146         w4 ← CEIL(w4*3)
147         w5 ← CEIL(w5*4)+2
148         w6 ← CEIL(w6*6)+6
149         IF  sbnr OF tmagv = 2
150             tmy ← ((CEIL(tmx/6))MODULO(2))*w6 + →
151             ((CEIL(tmx/6)+1)MODULO(2))*((w5)+((w2-1)*10))
152         END
153
154         IF sbnr OF tmagv =1 | sbnr OF tmagv = 3
155             tmy ← w3+6-(sbnr OF tmagv)MODULO(3)*10
156         END
157
158         IF cont20ft OF tmagv= TRUE
159             IF (agvnumber OF tmagv)MODULO(2)=1  @bovenlangs
160                 y OF tmagv ← ((tmy)-1)*10 + →
161                 (FLOOR((w2*0.5)*(((tmy)+1)MODULO(2)+3)))
162             END
163
164             IF (agvnumber OF tmagv)MODULO(2)=0  @onderlangs
165                 y OF tmagv ← ((tmy)-1)*10 + →
166                 (FLOOR((w2*0.5)*((tmy)MODULO(2)+3)))
167             END
168         GOTO almostready
169         END
170
171         IF cont20ft OF tmagv = FALSE
172             IF (agvnumber OF tmagv)MODULO(2)=1  @bovenlangs
173                 y OF tmagv ← ((tmy)-1)*10 + →
174                 ((2*w4)+(((tmy)-1)MODULO(2))+3)
175             END
176
177             IF (agvnumber OF tmagv)MODULO(2)=0  @onderlangs
178                 y OF tmagv ← ((tmy)-1)*10 + →
179                 ((2*w4)+((tmy)MODULO(2))+3)
180             END
181
182             GOTO almostready
183         END
184     END
185 almostready:
186
```

In **line 190 to 230** the rgv is located in a stack, not behind its own berth and has to go back to a stack behind its own berth

```
187 GOTO ready
188 END
189
190 @@@@
```

y creation chooses the cranes that have to serve the stacks. It is possible via the y_creation to vary the number of stacking cranes

```
 1
 2 begin:
 3     yyy OF tmagv←y OF tmagv
 4     xxx OF tmagv←FLOOR((yyy OF tmagv-1)/20)
 5     zzz OF tmagv←FLOOR((yyy OF tmagv-1)/20)
 6     yyy OF tmagv←(y OF tmagv)-(zzz OF tmagv)*20
 7
 8     IF (yyy OF tmagv)MODULO(2)=0
 9         yyy OF tmagv ← yyy OF tmagv -1
10     END
11     @IF (yyy OF tmagv)>10
12         @yyy OF tmagv← yyy OF tmagv -10
13     @END
14     yyy OF tmagv ← yyy OF tmagv + 20*(xxx OF tmagv)
15     agvcrane OF tmagv← FIRST crane IN cranelist→
16      WITH cranenumber = yyy OF tmagv
17 RETURN
```

Create outportations is a macro that creates the outportations

```
 1 i←1
 2 WHILE i≤300
 3     outportation[ i]←NEW DATASTREAM
 4     i←i+1
 5 END
 6
 7 ATTACH "outportation1.txt" TO outportation[ 1]
 8 ATTACH "outportation2.txt" TO outportation[ 2]
 9 ATTACH "outportation3.txt" TO outportation[ 3]
10 ATTACH "outportation4.txt" TO outportation[ 4]
11 ATTACH "outportation5.txt" TO outportation[ 5]
12 ATTACH "outportation6.txt" TO outportation[ 6]
13 ATTACH "outportation7.txt" TO outportation[ 7]
14 ATTACH "outportation8.txt" TO outportation[ 8]
15 ATTACH "outportation9.txt" TO outportation[ 9]
16 ATTACH "outportation10.txt" TO outportation[ 10]
17 ATTACH "outportation11.txt" TO outportation[ 11]
18 ATTACH "outportation12.txt" TO outportation[ 12]
19 ATTACH "outportation13.txt" TO outportation[ 13]
20 ATTACH "outportation14.txt" TO outportation[ 14]
21 ATTACH "outportation15.txt" TO outportation[ 15]
22 ATTACH "outportation16.txt" TO outportation[ 16]
23 ATTACH "outportation17.txt" TO outportation[ 17]
24 ATTACH "outportation18.txt" TO outportation[ 18]
25 ATTACH "outportation19.txt" TO outportation[ 19]
26 ATTACH "outportation20.txt" TO outportation[ 20]
27 ATTACH "outportation21.txt" TO outportation[ 21]
28 ATTACH "outportation22.txt" TO outportation[ 22]
29 ATTACH "outportation23.txt" TO outportation[ 23]
30 ATTACH "outportation24.txt" TO outportation[ 24]
31 ATTACH "outportation25.txt" TO outportation[ 25]
32 ATTACH "outportation26.txt" TO outportation[ 26]
33 ATTACH "outportation27.txt" TO outportation[ 27]
34 ATTACH "outportation28.txt" TO outportation[ 28]
35 ATTACH "outportation29.txt" TO outportation[ 29]
36 ATTACH "outportation30.txt" TO outportation[ 30]
37 ATTACH "outportation31.txt" TO outportation[ 31]
38 ATTACH "outportation32.txt" TO outportation[ 32]
39 ATTACH "outportation33.txt" TO outportation[ 33]
40 ATTACH "outportation34.txt" TO outportation[ 34]
41 ATTACH "outportation35.txt" TO outportation[ 35]
42 ATTACH "outportation36.txt" TO outportation[ 36]
43 ATTACH "outportation37.txt" TO outportation[ 37]
44 ATTACH "outportation38.txt" TO outportation[ 38]
45 ATTACH "outportation39.txt" TO outportation[ 39]
46 ATTACH "outportation40.txt" TO outportation[ 40]
47 ATTACH "outportation41.txt" TO outportation[ 41]
48 ATTACH "outportation42.txt" TO outportation[ 42]
49 ATTACH "outportation43.txt" TO outportation[ 43]
50 ATTACH "outportation44.txt" TO outportation[ 44]
51 ATTACH "outportation45.txt" TO outportation[ 45]
52 ATTACH "outportation46.txt" TO outportation[ 46]
53 ATTACH "outportation47.txt" TO outportation[ 47]
54 ATTACH "outportation48.txt" TO outportation[ 48]
55 ATTACH "outportation49.txt" TO outportation[ 49]
56 ATTACH "outportation50.txt" TO outportation[ 50]
57 ATTACH "outportation51.txt" TO outportation[ 51]
58 ATTACH "outportation52.txt" TO outportation[ 52]
59 ATTACH "outportation53.txt" TO outportation[ 53]
60 ATTACH "outportation54.txt" TO outportation[ 54]
61 ATTACH "outportation55.txt" TO outportation[ 55]
62 ATTACH "outportation56.txt" TO outportation[ 56]
```

```
 63 ATTACH "outportation57.txt" TO outportation[ 57]
 64 ATTACH "outportation58.txt" TO outportation[ 58]
 65 ATTACH "outportation59.txt" TO outportation[ 59]
 66 ATTACH "outportation60.txt" TO outportation[ 60]
 67 ATTACH "outportation61.txt" TO outportation[ 61]
 68 ATTACH "outportation62.txt" TO outportation[ 62]
 69 ATTACH "outportation63.txt" TO outportation[ 63]
 70 ATTACH "outportation64.txt" TO outportation[ 64]
 71 ATTACH "outportation65.txt" TO outportation[ 65]
 72 ATTACH "outportation66.txt" TO outportation[ 66]
 73 ATTACH "outportation67.txt" TO outportation[ 67]
 74 ATTACH "outportation68.txt" TO outportation[ 68]
 75 ATTACH "outportation69.txt" TO outportation[ 69]
 76 ATTACH "outportation70.txt" TO outportation[ 70]
 77 ATTACH "outportation71.txt" TO outportation[ 71]
 78 ATTACH "outportation72.txt" TO outportation[ 72]
 79 ATTACH "outportation73.txt" TO outportation[ 73]
 80 ATTACH "outportation74.txt" TO outportation[ 74]
 81 ATTACH "outportation75.txt" TO outportation[ 75]
 82 ATTACH "outportation76.txt" TO outportation[ 76]
 83 ATTACH "outportation77.txt" TO outportation[ 77]
 84 ATTACH "outportation78.txt" TO outportation[ 78]
 85 ATTACH "outportation79.txt" TO outportation[ 79]
 86 ATTACH "outportation80.txt" TO outportation[ 80]
 87 ATTACH "outportation81.txt" TO outportation[ 81]
 88 ATTACH "outportation82.txt" TO outportation[ 82]
 89 ATTACH "outportation83.txt" TO outportation[ 83]
 90 ATTACH "outportation84.txt" TO outportation[ 84]
 91 ATTACH "outportation85.txt" TO outportation[ 85]
 92 ATTACH "outportation86.txt" TO outportation[ 86]
 93 ATTACH "outportation87.txt" TO outportation[ 87]
 94 ATTACH "outportation88.txt" TO outportation[ 88]
 95 ATTACH "outportation89.txt" TO outportation[ 89]
 96 ATTACH "outportation90.txt" TO outportation[ 90]
 97 ATTACH "outportation91.txt" TO outportation[ 91]
 98 ATTACH "outportation92.txt" TO outportation[ 92]
 99 ATTACH "outportation93.txt" TO outportation[ 93]
100 ATTACH "outportation94.txt" TO outportation[ 94]
101 ATTACH "outportation95.txt" TO outportation[ 95]
102 ATTACH "outportation96.txt" TO outportation[ 96]
103 ATTACH "outportation97.txt" TO outportation[ 97]
104 ATTACH "outportation98.txt" TO outportation[ 98]
105 ATTACH "outportation99.txt" TO outportation[ 99]
106 ATTACH "outportation100.txt" TO outportation[ 100]
107 ATTACH "outportation101.txt" TO outportation[ 101]
108 ATTACH "outportation102.txt" TO outportation[ 102]
109 ATTACH "outportation103.txt" TO outportation[ 103]
110 ATTACH "outportation104.txt" TO outportation[ 104]
111 ATTACH "outportation105.txt" TO outportation[ 105]
112 ATTACH "outportation106.txt" TO outportation[ 106]
113 ATTACH "outportation107.txt" TO outportation[ 107]
114 ATTACH "outportation108.txt" TO outportation[ 108]
115 ATTACH "outportation109.txt" TO outportation[ 109]
116 ATTACH "outportation110.txt" TO outportation[ 110]
117 ATTACH "outportation111.txt" TO outportation[ 111]
118 ATTACH "outportation112.txt" TO outportation[ 112]
119 ATTACH "outportation113.txt" TO outportation[ 113]
120 ATTACH "outportation114.txt" TO outportation[ 114]
121 ATTACH "outportation115.txt" TO outportation[ 115]
122 ATTACH "outportation116.txt" TO outportation[ 116]
123 ATTACH "outportation117.txt" TO outportation[ 117]
124 ATTACH "outportation118.txt" TO outportation[ 118]
```

```
125 ATTACH "outportation119.txt" TO outportation[ 119]
126 ATTACH "outportation120.txt" TO outportation[ 120]
127 ATTACH "outportation121.txt" TO outportation[ 121]
128 ATTACH "outportation122.txt" TO outportation[ 122]
129 ATTACH "outportation123.txt" TO outportation[ 123]
130 ATTACH "outportation124.txt" TO outportation[ 124]
131 ATTACH "outportation125.txt" TO outportation[ 125]
132 ATTACH "outportation126.txt" TO outportation[ 126]
133 ATTACH "outportation127.txt" TO outportation[ 127]
134 ATTACH "outportation128.txt" TO outportation[ 128]
135 ATTACH "outportation129.txt" TO outportation[ 129]
136 ATTACH "outportation130.txt" TO outportation[ 130]
137 ATTACH "outportation131.txt" TO outportation[ 131]
138 ATTACH "outportation132.txt" TO outportation[ 132]
139 ATTACH "outportation133.txt" TO outportation[ 133]
140 ATTACH "outportation134.txt" TO outportation[ 134]
141 ATTACH "outportation135.txt" TO outportation[ 135]
142 ATTACH "outportation136.txt" TO outportation[ 136]
143 ATTACH "outportation137.txt" TO outportation[ 137]
144 ATTACH "outportation138.txt" TO outportation[ 138]
145 ATTACH "outportation139.txt" TO outportation[ 139]
146 ATTACH "outportation140.txt" TO outportation[ 140]
147 ATTACH "outportation141.txt" TO outportation[ 141]
148 ATTACH "outportation142.txt" TO outportation[ 142]
149 ATTACH "outportation143.txt" TO outportation[ 143]
150 ATTACH "outportation144.txt" TO outportation[ 144]
151 ATTACH "outportation145.txt" TO outportation[ 145]
152 ATTACH "outportation146.txt" TO outportation[ 146]
153 ATTACH "outportation147.txt" TO outportation[ 147]
154 ATTACH "outportation148.txt" TO outportation[ 148]
155 ATTACH "outportation149.txt" TO outportation[ 149]
156 ATTACH "outportation150.txt" TO outportation[ 150]
157 ATTACH "outportation151.txt" TO outportation[ 151]
158 ATTACH "outportation152.txt" TO outportation[ 152]
159 ATTACH "outportation153.txt" TO outportation[ 153]
160 ATTACH "outportation154.txt" TO outportation[ 154]
161 ATTACH "outportation155.txt" TO outportation[ 155]
162 ATTACH "outportation156.txt" TO outportation[ 156]
163 ATTACH "outportation157.txt" TO outportation[ 157]
164 ATTACH "outportation158.txt" TO outportation[ 158]
165 ATTACH "outportation159.txt" TO outportation[ 159]
166 ATTACH "outportation160.txt" TO outportation[ 160]
167 ATTACH "outportation161.txt" TO outportation[ 161]
168 ATTACH "outportation162.txt" TO outportation[ 162]
169 ATTACH "outportation163.txt" TO outportation[ 163]
170 ATTACH "outportation164.txt" TO outportation[ 164]
171 ATTACH "outportation165.txt" TO outportation[ 165]
172 ATTACH "outportation166.txt" TO outportation[ 166]
173 ATTACH "outportation167.txt" TO outportation[ 167]
174 ATTACH "outportation168.txt" TO outportation[ 168]
175 ATTACH "outportation169.txt" TO outportation[ 169]
176 ATTACH "outportation170.txt" TO outportation[ 170]
177 ATTACH "outportation171.txt" TO outportation[ 171]
178 ATTACH "outportation172.txt" TO outportation[ 172]
179 ATTACH "outportation173.txt" TO outportation[ 173]
180 ATTACH "outportation174.txt" TO outportation[ 174]
181 ATTACH "outportation175.txt" TO outportation[ 175]
182 ATTACH "outportation176.txt" TO outportation[ 176]
183 ATTACH "outportation177.txt" TO outportation[ 177]
184 ATTACH "outportation178.txt" TO outportation[ 178]
185 ATTACH "outportation179.txt" TO outportation[ 179]
186 ATTACH "outportation180.txt" TO outportation[ 180]
```

```
187 ATTACH "outportation181.txt" TO outportation[ 181]
188 ATTACH "outportation182.txt" TO outportation[ 182]
189 ATTACH "outportation183.txt" TO outportation[ 183]
190 ATTACH "outportation184.txt" TO outportation[ 184]
191 ATTACH "outportation185.txt" TO outportation[ 185]
192 ATTACH "outportation186.txt" TO outportation[ 186]
193 ATTACH "outportation187.txt" TO outportation[ 187]
194 ATTACH "outportation188.txt" TO outportation[ 188]
195 ATTACH "outportation189.txt" TO outportation[ 189]
196 ATTACH "outportation190.txt" TO outportation[ 190]
197 ATTACH "outportation191.txt" TO outportation[ 191]
198 ATTACH "outportation192.txt" TO outportation[ 192]
199 ATTACH "outportation193.txt" TO outportation[ 193]
200 ATTACH "outportation194.txt" TO outportation[ 194]
201 ATTACH "outportation195.txt" TO outportation[ 195]
202 ATTACH "outportation196.txt" TO outportation[ 196]
203 ATTACH "outportation197.txt" TO outportation[ 197]
204 ATTACH "outportation198.txt" TO outportation[ 198]
205 ATTACH "outportation199.txt" TO outportation[ 199]
206 ATTACH "outportation200.txt" TO outportation[ 200]
207
208 ATTACH "outportation201.txt" TO outportation[ 201]
209 ATTACH "outportation202.txt" TO outportation[ 202]
210 ATTACH "outportation203.txt" TO outportation[ 203]
211 ATTACH "outportation204.txt" TO outportation[ 204]
212 ATTACH "outportation205.txt" TO outportation[ 205]
213 ATTACH "outportation206.txt" TO outportation[ 206]
214 ATTACH "outportation207.txt" TO outportation[ 207]
215 ATTACH "outportation208.txt" TO outportation[ 208]
216 ATTACH "outportation209.txt" TO outportation[ 209]
217 ATTACH "outportation210.txt" TO outportation[ 210]
218 ATTACH "outportation211.txt" TO outportation[ 211]
219 ATTACH "outportation212.txt" TO outportation[ 212]
220 ATTACH "outportation213.txt" TO outportation[ 213]
221 ATTACH "outportation214.txt" TO outportation[ 214]
222 ATTACH "outportation215.txt" TO outportation[ 215]
223 ATTACH "outportation216.txt" TO outportation[ 216]
224 ATTACH "outportation217.txt" TO outportation[ 217]
225 ATTACH "outportation218.txt" TO outportation[ 218]
226 ATTACH "outportation219.txt" TO outportation[ 219]
227 ATTACH "outportation220.txt" TO outportation[ 220]
228 ATTACH "outportation221.txt" TO outportation[ 221]
229 ATTACH "outportation222.txt" TO outportation[ 222]
230 ATTACH "outportation223.txt" TO outportation[ 223]
231 ATTACH "outportation224.txt" TO outportation[ 224]
232 ATTACH "outportation225.txt" TO outportation[ 225]
233 ATTACH "outportation226.txt" TO outportation[ 226]
234 ATTACH "outportation227.txt" TO outportation[ 227]
235 ATTACH "outportation228.txt" TO outportation[ 228]
236 ATTACH "outportation229.txt" TO outportation[ 229]
237 ATTACH "outportation230.txt" TO outportation[ 230]
238 ATTACH "outportation231.txt" TO outportation[ 231]
239 ATTACH "outportation232.txt" TO outportation[ 232]
240 ATTACH "outportation233.txt" TO outportation[ 233]
241 ATTACH "outportation234.txt" TO outportation[ 234]
242 ATTACH "outportation235.txt" TO outportation[ 235]
243 ATTACH "outportation236.txt" TO outportation[ 236]
244 ATTACH "outportation237.txt" TO outportation[ 237]
245 ATTACH "outportation238.txt" TO outportation[ 238]
246 ATTACH "outportation239.txt" TO outportation[ 239]
247 ATTACH "outportation240.txt" TO outportation[ 240]
248 ATTACH "outportation241.txt" TO outportation[ 241]
```

```
249 ATTACH "outportation242.txt" TO outportation[ 242]
250 ATTACH "outportation243.txt" TO outportation[ 243]
251 ATTACH "outportation244.txt" TO outportation[ 244]
252 ATTACH "outportation245.txt" TO outportation[ 245]
253 ATTACH "outportation246.txt" TO outportation[ 246]
254 ATTACH "outportation247.txt" TO outportation[ 247]
255 ATTACH "outportation248.txt" TO outportation[ 248]
256 ATTACH "outportation249.txt" TO outportation[ 249]
257 ATTACH "outportation250.txt" TO outportation[ 250]
258 ATTACH "outportation251.txt" TO outportation[ 251]
259 ATTACH "outportation252.txt" TO outportation[ 252]
260 ATTACH "outportation253.txt" TO outportation[ 253]
261 ATTACH "outportation254.txt" TO outportation[ 254]
262 ATTACH "outportation255.txt" TO outportation[ 255]
263 ATTACH "outportation256.txt" TO outportation[ 256]
264 ATTACH "outportation257.txt" TO outportation[ 257]
265 ATTACH "outportation258.txt" TO outportation[ 258]
266 ATTACH "outportation259.txt" TO outportation[ 259]
267 ATTACH "outportation260.txt" TO outportation[ 260]
268 ATTACH "outportation261.txt" TO outportation[ 261]
269 ATTACH "outportation262.txt" TO outportation[ 262]
270 ATTACH "outportation263.txt" TO outportation[ 263]
271 ATTACH "outportation264.txt" TO outportation[ 264]
272 ATTACH "outportation265.txt" TO outportation[ 265]
273 ATTACH "outportation266.txt" TO outportation[ 266]
274 ATTACH "outportation267.txt" TO outportation[ 267]
275 ATTACH "outportation268.txt" TO outportation[ 268]
276 ATTACH "outportation269.txt" TO outportation[ 269]
277 ATTACH "outportation270.txt" TO outportation[ 270]
278 ATTACH "outportation271.txt" TO outportation[ 271]
279 ATTACH "outportation272.txt" TO outportation[ 272]
280 ATTACH "outportation273.txt" TO outportation[ 273]
281 ATTACH "outportation274.txt" TO outportation[ 274]
282 ATTACH "outportation275.txt" TO outportation[ 275]
283 ATTACH "outportation276.txt" TO outportation[ 276]
284 ATTACH "outportation277.txt" TO outportation[ 277]
285 ATTACH "outportation278.txt" TO outportation[ 278]
286 ATTACH "outportation279.txt" TO outportation[ 279]
287 ATTACH "outportation280.txt" TO outportation[ 280]
288 ATTACH "outportation281.txt" TO outportation[ 281]
289 ATTACH "outportation282.txt" TO outportation[ 282]
290 ATTACH "outportation283.txt" TO outportation[ 283]
291 ATTACH "outportation284.txt" TO outportation[ 284]
292 ATTACH "outportation285.txt" TO outportation[ 285]
293 ATTACH "outportation286.txt" TO outportation[ 286]
294 ATTACH "outportation287.txt" TO outportation[ 287]
295 ATTACH "outportation288.txt" TO outportation[ 288]
296 ATTACH "outportation289.txt" TO outportation[ 289]
297 ATTACH "outportation290.txt" TO outportation[ 290]
298 ATTACH "outportation291.txt" TO outportation[ 291]
299 ATTACH "outportation292.txt" TO outportation[ 292]
300 ATTACH "outportation293.txt" TO outportation[ 293]
301 ATTACH "outportation294.txt" TO outportation[ 294]
302 ATTACH "outportation295.txt" TO outportation[ 295]
303 ATTACH "outportation296.txt" TO outportation[ 296]
304 ATTACH "outportation297.txt" TO outportation[ 297]
305 ATTACH "outportation298.txt" TO outportation[ 298]
306 ATTACH "outportation299.txt" TO outportation[ 299]
307 ATTACH "outportation300.txt" TO outportation[ 300]
308 RETURN
```

crane cycletime is a macro that creates the cycletimes for the stacking cranes

F is the time in seconds the crane has to work

```
 1 @this is where the cycle times of the stacking cranes are created
 2
 3 begin:
 4 w1←SAMPLE FROM unif
 5 w2←SAMPLE FROM UNIF
 6      IF w1≥0.5
 7          w1←80+36*w2
 8      END
 9      IF w1<0.5
10          w1←44 +36*w2
11      END
12 F OF THIS CRANE ←W1
13
14 RETURN
```

crane outportations is a macro that creates the outportations for the stacking cranes every crane gets its own outportation if it has been doing something

```
 1 G←CRANENUMBER OF THIS CRANE
 2
 3          CONVERT (cranenumber OF THIS crane) TO outstring FIELDLENGTH 12
 4          WRITE OUTSTRING TO OUTPORTATION[ G] WITH IMAGE a(12)
 5          CONVERT NOW TO OUTSTRING FIELDLENGTH 12 DECIMALS 3
 6          WRITE outstring TO outportation[ G] WITH IMAGE A(12)
 7          CONVERT f OF THIS crane TO OUTSTRING FIELDLENGTH 12 DECIMALS 3
 8          WRITE outstring TO outportation[ G] WITH IMAGE A(12)|
 9
10 RETURN
```

creating sections is a macro that creates sections and attach these sections to windws in this way all the queues at the sections can be monitored

```
 1 newsection←FIRST section IN sections WITH sec_row_number = mn
 2 secrow[ 1]←NEW WINDOW CALLED "noderow1"
 3 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 1]
 4
 5 newsection←FIRST section IN sections WITH sec_row_number = mnb
 6 secrow[ 2]←NEW WINDOW CALLED "noderow2"
 7 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 2]
 8
 9 newsection←FIRST section IN sections WITH sec_row_number = bmn
10 secrow[ 3]←NEW WINDOW CALLED "noderow3"
11 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 3]
12
13 newsection←FIRST section IN sections WITH sec_row_number = nm
14 secrow[ 4]←NEW WINDOW CALLED "noderow4"
15 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 4]
16
17 newsection←FIRST section IN sections WITH sec_row_number = 116
18 secrow[ 145]←NEW WINDOW CALLED "noderow116"
19 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 145]
20
21 newsection←FIRST section IN sections WITH sec_row_number = 114
22 secrow[ 146]←NEW WINDOW CALLED "noderow114"
23 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 146]
24
25 newsection←FIRST section IN sections WITH sec_row_number = 112
26 secrow[ 147]←NEW WINDOW CALLED "noderow112"
27 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 147]
28
29 newsection←FIRST section IN sections WITH sec_row_number = 110
30 secrow[ 148]←NEW WINDOW CALLED "noderow110"
31 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 148]
32
33 newsection←FIRST section IN sections WITH sec_row_number = 108
34 secrow[ 149]←NEW WINDOW CALLED "noderow108"
35 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 149]
36
37 newsection←FIRST section IN sections WITH sec_row_number = 106
38 secrow[ 150]←NEW WINDOW CALLED "noderow106"
39 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 150]
40
41 newsection←FIRST section IN sections WITH sec_row_number = 104
42 secrow[ 151]←NEW WINDOW CALLED "noderow104"
43 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 151]
44
45 newsection←FIRST section IN sections WITH sec_row_number = 102
46 secrow[ 152]←NEW WINDOW CALLED "noderow102"
47 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 152]
48
49 newsection←FIRST section IN sections WITH sec_row_number = 260
50 secrow[ 153]  ←NEW WINDOW CALLED "noderow260"
51 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 153]
52
53 newsection←FIRST section IN sections WITH sec_row_number = 290
54 secrow[ 154]←NEW WINDOW CALLED "noderow290"
55 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 154]
56
57 newsection←FIRST section IN sections WITH sec_row_number = 320
58 secrow[ 155]←NEW WINDOW CALLED "noderow320"
59 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 155]
60
61 newsection←FIRST section IN sections WITH sec_row_number = 350
62 secrow[ 156]←NEW WINDOW CALLED "noderow350"
```

```
ATTACH HISTORY OF sec_row OF newsection TO secrow[156]

newsection←FIRST section IN sections WITH sec_row_number = 380
secrow[157]←NEW WINDOW CALLED "noderow380"
ATTACH HISTORY OF sec_row OF newsection TO secrow[157]

newsection←FIRST section IN sections WITH sec_row_number = 410
secrow[158]←NEW WINDOW CALLED "noderow410"
ATTACH HISTORY OF sec_row OF newsection TO secrow[158]

newsection←FIRST section IN sections WITH sec_row_number = 440
secrow[159]←NEW WINDOW CALLED "noderow440"
ATTACH HISTORY OF sec_row OF newsection TO secrow[159]


newsection←FIRST section IN sections WITH sec_row_number = 535
secrow[160]←NEW WINDOW CALLED "noderow535"
ATTACH HISTORY OF sec_row OF newsection TO secrow[160]

newsection←FIRST section IN sections WITH sec_row_number = 533
secrow[161]←NEW WINDOW CALLED "noderow533"
ATTACH HISTORY OF sec_row OF newsection TO secrow[161]

newsection←FIRST section IN sections WITH sec_row_number = 531
secrow[162]←NEW WINDOW CALLED "noderow531"
ATTACH HISTORY OF sec_row OF newsection TO secrow[162]

newsection←FIRST section IN sections WITH sec_row_number = 504
secrow[163]←NEW WINDOW CALLED "noderow504"
ATTACH HISTORY OF sec_row OF newsection TO secrow[163]

newsection←FIRST section IN sections WITH sec_row_number = 506
secrow[164]←NEW WINDOW CALLED "noderow506"
ATTACH HISTORY OF sec_row OF newsection TO secrow[164]

newsection←FIRST section IN sections WITH sec_row_number = 508
secrow[165]←NEW WINDOW CALLED "noderow508"
ATTACH HISTORY OF sec_row OF newsection TO secrow[165]

newsection←FIRST section IN sections WITH sec_row_number = 510
secrow[166]←NEW WINDOW CALLED "noderow510"
ATTACH HISTORY OF sec_row OF newsection TO secrow[166]

newsection←FIRST section IN sections WITH sec_row_number = 512
secrow[167]←NEW WINDOW CALLED "noderow512"
ATTACH HISTORY OF sec_row OF newsection TO secrow[167]


newsection←FIRST section IN sections WITH sec_row_number = 514
secrow[168]←NEW WINDOW CALLED "noderow514"
ATTACH HISTORY OF sec_row OF newsection TO secrow[168]

newsection←FIRST section IN sections WITH sec_row_number = 516
secrow[169]←NEW WINDOW CALLED "noderow516"
ATTACH HISTORY OF sec_row OF newsection TO secrow[169]

newsection←FIRST section IN sections WITH sec_row_number = 243
secrow[170]←NEW WINDOW CALLED "noderow243"
ATTACH HISTORY OF sec_row OF newsection TO secrow[170]

newsection←FIRST section IN sections WITH sec_row_number = 273
secrow[171]←NEW WINDOW CALLED "noderow273"
```

```
125 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 171]
126
127 newsection←FIRST section IN sections WITH sec_row_number = 303
128 secrow[ 172] ←NEW WINDOW CALLED "noderow303"
129 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 172]
130
131 newsection←FIRST section IN sections WITH sec_row_number = 333
132 secrow[ 173] ←NEW WINDOW CALLED "noderow333"
133 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 173]
134
135 newsection←FIRST section IN sections WITH sec_row_number = 363
136 secrow[ 174] ←NEW WINDOW CALLED "noderow363"
137 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 174]
138
139 newsection←FIRST section IN sections WITH sec_row_number = 393
140 secrow[ 175] ←NEW WINDOW CALLED "noderow393"
141 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 175]
142
143 newsection←FIRST section IN sections WITH sec_row_number = 423
144 secrow[ 176] ←NEW WINDOW CALLED "noderow423"
145 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 176]
146
147 newsection←FIRST section IN sections WITH sec_row_number = 453
148 secrow[ 177] ←NEW WINDOW CALLED "noderow453"
149 ATTACH HISTORY OF sec_row OF newsection TO secrow[ 177]
150
151
152
153 @lekker met de stacks bezig joh zijn er niet zo veel
154
155 newstack←FIRST stack IN stacklist WITH stacknumber = c1
156 secrow[ 5] ←NEW WINDOW CALLED "stack21"
157 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 5]
158
159 newstack←FIRST stack IN stacklist WITH stacknumber = c2
160 secrow[ 6] ←NEW WINDOW CALLED "stack22"
161 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 6]
162
163 newstack←FIRST stack IN stacklist WITH stacknumber = c3
164 secrow[ 7] ←NEW WINDOW CALLED "stack23"
165 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 7]
166
167 newstack←FIRST stack IN stacklist WITH stacknumber = c4
168 secrow[ 8] ←NEW WINDOW CALLED "stack24"
169 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 8]
170
171 newstack←FIRST stack IN stacklist WITH stacknumber = c5
172 secrow[ 9] ←NEW WINDOW CALLED "stack25"
173 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 9]
174
175 newstack←FIRST stack IN stacklist WITH stacknumber = c6
176 secrow[ 10] ←NEW WINDOW CALLED "stack26"
177 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 10]
178
179 newstack←FIRST stack IN stacklist WITH stacknumber = c7
180 secrow[ 11] ←NEW WINDOW CALLED "stack27"
181 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 11]
182
183 newstack←FIRST stack IN stacklist WITH stacknumber = c8
184 secrow[ 12] ←NEW WINDOW CALLED "stack28"
185 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 12]
186
```

```
187 newstack←FIRST stack IN stacklist WITH stacknumber = c9
188 secrow[ 13]←NEW WINDOW CALLED "stack29"
189 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 13]
190
191 newstack←FIRST stack IN stacklist WITH stacknumber = c10
192 secrow[ 14]←NEW WINDOW CALLED "stack30"
193 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 14]
194
195 newstack←FIRST stack IN stacklist WITH stacknumber = 31
196 secrow[ 15]←NEW WINDOW CALLED "stack31"
197 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 15]
198
199 newstack←FIRST stack IN stacklist WITH stacknumber = 32
200 secrow[ 16]←NEW WINDOW CALLED "stack32"
201 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 16]
202
203 newstack←FIRST stack IN stacklist WITH stacknumber = 33
204 secrow[ 17]←NEW WINDOW CALLED "stack33"
205 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 17]
206
207 newstack←FIRST stack IN stacklist WITH stacknumber = 34
208 secrow[ 18]←NEW WINDOW CALLED "stack34"
209 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 18]
210
211 newstack←FIRST stack IN stacklist WITH stacknumber = 35
212 secrow[ 19]←NEW WINDOW CALLED "stack35"
213 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 19]
214
215 newstack←FIRST stack IN stacklist WITH stacknumber = 36
216 secrow[ 20]←NEW WINDOW CALLED "stack36"
217 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 20]
218
219 newstack←FIRST stack IN stacklist WITH stacknumber = 37
220 secrow[ 21]←NEW WINDOW CALLED "stack37"
221 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 21]
222
223 newstack←FIRST stack IN stacklist WITH stacknumber = 38
224 secrow[ 22]←NEW WINDOW CALLED "stack38"
225 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 22]
226
227 newstack←FIRST stack IN stacklist WITH stacknumber = 39
228 secrow[ 23]←NEW WINDOW CALLED "stack39"
229 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 23]
230
231 newstack←FIRST stack IN stacklist WITH stacknumber = 40
232 secrow[ 24]←NEW WINDOW CALLED "stack40"
233 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 24]
234
235 newstack←FIRST stack IN stacklist WITH stacknumber = 41
236 secrow[ 25]←NEW WINDOW CALLED "stack41"
237 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 25]
238
239 newstack←FIRST stack IN stacklist WITH stacknumber = 42
240 secrow[ 26]←NEW WINDOW CALLED "stack42"
241 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 26]
242
243 newstack←FIRST stack IN stacklist WITH stacknumber = 43
244 secrow[ 27]←NEW WINDOW CALLED "stack43"
245 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 27]
246
247 newstack←FIRST stack IN stacklist WITH stacknumber = 44
248 secrow[ 28]←NEW WINDOW CALLED "stack44"
```

```
249 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 28]
250
251 newstack←FIRST stack IN stacklist WITH stacknumber = 45
252 secrow[ 29]←NEW WINDOW CALLED "stack45"
253 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 29]
254
255 newstack←FIRST stack IN stacklist WITH stacknumber = 46
256 secrow[ 30]←NEW WINDOW CALLED "stack46"
257 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 30]
258
259 newstack←FIRST stack IN stacklist WITH stacknumber = 47
260 secrow[ 31]←NEW WINDOW CALLED "stack47"
261 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 31]
262
263 newstack←FIRST stack IN stacklist WITH stacknumber = 48
264 secrow[ 32]←NEW WINDOW CALLED "stack48"
265 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 32]
266
267 newstack←FIRST stack IN stacklist WITH stacknumber = 49
268 secrow[ 33]←NEW WINDOW CALLED "stack49"
269 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 33]
270
271 newstack←FIRST stack IN stacklist WITH stacknumber = 50
272 secrow[ 34]←NEW WINDOW CALLED "stack50"
273 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 34]
274
275 newstack←FIRST stack IN stacklist WITH stacknumber = 51
276 secrow[ 35]←NEW WINDOW CALLED "stack51"
277 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 35]
278
279 newstack←FIRST stack IN stacklist WITH stacknumber = 52
280 secrow[ 36]←NEW WINDOW CALLED "stack52"
281 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 36]
282
283 newstack←FIRST stack IN stacklist WITH stacknumber = 53
284 secrow[ 37]←NEW WINDOW CALLED "stack53"
285 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 37]
286
287 newstack←FIRST stack IN stacklist WITH stacknumber = 54
288 secrow[ 38]←NEW WINDOW CALLED "stack54"
289 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 38]
290
291 newstack←FIRST stack IN stacklist WITH stacknumber = 55
292 secrow[ 39]←NEW WINDOW CALLED "stack55"
293 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 39]
294
295 newstack←FIRST stack IN stacklist WITH stacknumber = 56
296 secrow[ 40]←NEW WINDOW CALLED "stack56"
297 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 40]
298
299 newstack←FIRST stack IN stacklist WITH stacknumber = 57
300 secrow[ 41]←NEW WINDOW CALLED "stack57"
301 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 41]
302
303 newstack←FIRST stack IN stacklist WITH stacknumber = 58
304 secrow[ 42]←NEW WINDOW CALLED "stack58"
305 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 42]
306
307 newstack←FIRST stack IN stacklist WITH stacknumber = 59
308 secrow[ 43]←NEW WINDOW CALLED "stack59"
309 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 43]
310
```

```
311 newstack←FIRST stack IN stacklist WITH stacknumber = 60
312 secrow[ 44]←NEW WINDOW CALLED "stack60"
313 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 44]
314
315 newstack←FIRST stack IN stacklist WITH stacknumber = 61
316 secrow[ 45]←NEW WINDOW CALLED "stack61"
317 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 45]
318
319 newstack←FIRST stack IN stacklist WITH stacknumber = 62
320 secrow[ 46]←NEW WINDOW CALLED "stack62"
321 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 46]
322
323 newstack←FIRST stack IN stacklist WITH stacknumber = 63
324 secrow[ 47]←NEW WINDOW CALLED "stack63"
325 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 47]
326
327 newstack←FIRST stack IN stacklist WITH stacknumber = 64
328 secrow[ 48]←NEW WINDOW CALLED "stack64"
329 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 48]
330
331 newstack←FIRST stack IN stacklist WITH stacknumber = 65
332 secrow[ 49]←NEW WINDOW CALLED "stack65"
333 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 49]
334
335 newstack←FIRST stack IN stacklist WITH stacknumber = 66
336 secrow[ 50]←NEW WINDOW CALLED "stack66"
337 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 50]
338
339 newstack←FIRST stack IN stacklist WITH stacknumber = 67
340 secrow[ 51]←NEW WINDOW CALLED "stack67"
341 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 51]
342
343 newstack←FIRST stack IN stacklist WITH stacknumber = 68
344 secrow[ 52]←NEW WINDOW CALLED "stack68"
345 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 52]
346
347 newstack←FIRST stack IN stacklist WITH stacknumber = 69
348 secrow[ 53]←NEW WINDOW CALLED "stack69"
349 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 53]
350
351 newstack←FIRST stack IN stacklist WITH stacknumber = 70
352 secrow[ 54]←NEW WINDOW CALLED "stack70"
353 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 54]
354
355 newstack←FIRST stack IN stacklist WITH stacknumber = 71
356 secrow[ 55]←NEW WINDOW CALLED "stack71"
357 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 55]
358
359 newstack←FIRST stack IN stacklist WITH stacknumber = 72
360 secrow[ 56]←NEW WINDOW CALLED "stack72"
361 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 56]
362
363 newstack←FIRST stack IN stacklist WITH stacknumber = 73
364 secrow[ 57]←NEW WINDOW CALLED "stack73"
365 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 57]
366
367 newstack←FIRST stack IN stacklist WITH stacknumber = 74
368 secrow[ 58]←NEW WINDOW CALLED "stack74"
369 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 58]
370
371 newstack←FIRST stack IN stacklist WITH stacknumber = 75
372 secrow[ 59]←NEW WINDOW CALLED "stack75"
```

```
373 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 59]
374
375 newstack←FIRST stack IN stacklist WITH stacknumber = 76
376 secrow[ 60]←NEW WINDOW CALLED "stack76"
377 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 60]
378
379 newstack←FIRST stack IN stacklist WITH stacknumber = 77
380 secrow[ 61]←NEW WINDOW CALLED "stack77"
381 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 61]
382
383 newstack←FIRST stack IN stacklist WITH stacknumber = 78
384 secrow[ 62]←NEW WINDOW CALLED "stack78"
385 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 62]
386
387 newstack←FIRST stack IN stacklist WITH stacknumber = 79
388 secrow[ 63]←NEW WINDOW CALLED "stack79"
389 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 63]
390
391 newstack←FIRST stack IN stacklist WITH stacknumber = 80
392 secrow[ 64]←NEW WINDOW CALLED "stack80"
393 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 64]
394
395 newstack←FIRST stack IN stacklist WITH stacknumber = 81
396 secrow[ 65]←NEW WINDOW CALLED "stack81"
397 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 65]
398
399 newstack←FIRST stack IN stacklist WITH stacknumber = 82
400 secrow[ 66]←NEW WINDOW CALLED "stack82"
401 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 66]
402
403 newstack←FIRST stack IN stacklist WITH stacknumber = 83
404 secrow[ 67]←NEW WINDOW CALLED "stack83"
405 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 67]
406
407 newstack←FIRST stack IN stacklist WITH stacknumber = 84
408 secrow[ 68]←NEW WINDOW CALLED "stack84"
409 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 68]
410
411 newstack←FIRST stack IN stacklist WITH stacknumber = 85
412 secrow[ 69]←NEW WINDOW CALLED "stack85"
413 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 69]
414
415 newstack←FIRST stack IN stacklist WITH stacknumber = 86
416 secrow[ 70]←NEW WINDOW CALLED "stack86"
417 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 70]
418
419 newstack←FIRST stack IN stacklist WITH stacknumber = 87
420 secrow[ 71]←NEW WINDOW CALLED "stack87"
421 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 71]
422
423 newstack←FIRST stack IN stacklist WITH stacknumber = 88
424 secrow[ 72]←NEW WINDOW CALLED "stack88"
425 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 72]
426
427 newstack←FIRST stack IN stacklist WITH stacknumber = 89
428 secrow[ 73]←NEW WINDOW CALLED "stack89"
429 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 73]
430
431 newstack←FIRST stack IN stacklist WITH stacknumber = 90
432 secrow[ 74]←NEW WINDOW CALLED "stack90"
433 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 74]
434
```

```
435 newstack←FIRST stack IN stacklist WITH stacknumber = 91
436 secrow[ 75]←NEW WINDOW CALLED "stack91"
437 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 75]
438
439 newstack←FIRST stack IN stacklist WITH stacknumber = 92
440 secrow[ 76]←NEW WINDOW CALLED "stack92"
441 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 76]
442
443 newstack←FIRST stack IN stacklist WITH stacknumber = 93
444 secrow[ 77]←NEW WINDOW CALLED "stack93"
445 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 77]
446
447 newstack←FIRST stack IN stacklist WITH stacknumber = 94
448 secrow[ 78]←NEW WINDOW CALLED "stack94"
449 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 78]
450
451 newstack←FIRST stack IN stacklist WITH stacknumber = 95
452 secrow[ 79]←NEW WINDOW CALLED "stack95"
453 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 79]
454
455 newstack←FIRST stack IN stacklist WITH stacknumber = 96
456 secrow[ 80]←NEW WINDOW CALLED "stack96"
457 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 80]
458
459 newstack←FIRST stack IN stacklist WITH stacknumber = 97
460 secrow[ 81]←NEW WINDOW CALLED "stack97"
461 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 81]
462
463 newstack←FIRST stack IN stacklist WITH stacknumber = 98
464 secrow[ 82]←NEW WINDOW CALLED "stack98"
465 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 82]
466
467 newstack←FIRST stack IN stacklist WITH stacknumber = 99
468 secrow[ 83]←NEW WINDOW CALLED "stack99"
469 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 83]
470
471 newstack←FIRST stack IN stacklist WITH stacknumber = 100
472 secrow[ 84]←NEW WINDOW CALLED "stack100"
473 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 84]
474
475 newstack←FIRST stack IN stacklist WITH stacknumber = 101
476 secrow[ 85]←NEW WINDOW CALLED "stack101"
477 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 85]
478
479 newstack←FIRST stack IN stacklist WITH stacknumber = 102
480 secrow[ 86]←NEW WINDOW CALLED "stack102"
481 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 86]
482
483 newstack←FIRST stack IN stacklist WITH stacknumber = 103
484 secrow[ 87]←NEW WINDOW CALLED "stack103"
485 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 87]
486
487 newstack←FIRST stack IN stacklist WITH stacknumber = 104
488 secrow[ 88]←NEW WINDOW CALLED "stack104"
489 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 88]
490
491 newstack←FIRST stack IN stacklist WITH stacknumber = 105
492 secrow[ 89]←NEW WINDOW CALLED "stack105"
493 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 89]
494
495 newstack←FIRST stack IN stacklist WITH stacknumber = 106
496 secrow[ 90]←NEW WINDOW CALLED "stack106"
```

```
497 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 90]
498
499 newstack←FIRST stack IN stacklist WITH stacknumber = 107
500 secrow[ 91]←NEW WINDOW CALLED "stack107"
501 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 91]
502
503 newstack←FIRST stack IN stacklist WITH stacknumber = 108
504 secrow[ 92]←NEW WINDOW CALLED "stack108"
505 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 92]
506
507 newstack←FIRST stack IN stacklist WITH stacknumber = 109
508 secrow[ 93]←NEW WINDOW CALLED "stack109"
509 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 93]
510
511 newstack←FIRST stack IN stacklist WITH stacknumber = 110
512 secrow[ 94]←NEW WINDOW CALLED "stack110"
513 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 94]
514
515 newstack←FIRST stack IN stacklist WITH stacknumber = 111
516 secrow[ 95]←NEW WINDOW CALLED "stack111"
517 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 95]
518
519 newstack←FIRST stack IN stacklist WITH stacknumber = 112
520 secrow[ 96]←NEW WINDOW CALLED "stack112"
521 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 96]
522
523 newstack←FIRST stack IN stacklist WITH stacknumber = 113
524 secrow[ 97]←NEW WINDOW CALLED "stack113"
525 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 97]
526
527 newstack←FIRST stack IN stacklist WITH stacknumber = 114
528 secrow[ 98]←NEW WINDOW CALLED "stack114"
529 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 98]
530
531 newstack←FIRST stack IN stacklist WITH stacknumber = 115
532 secrow[ 99]←NEW WINDOW CALLED "stack115"
533 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 99]
534
535 newstack←FIRST stack IN stacklist WITH stacknumber = 116
536 secrow[ 100]←NEW WINDOW CALLED "stack116"
537 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 100]
538
539 newstack←FIRST stack IN stacklist WITH stacknumber = 117
540 secrow[ 101]←NEW WINDOW CALLED "stack117"
541 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 101]
542
543 newstack←FIRST stack IN stacklist WITH stacknumber = 118
544 secrow[ 102]←NEW WINDOW CALLED "stack118"
545 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 102]
546
547 newstack←FIRST stack IN stacklist WITH stacknumber =119
548 secrow[ 103]←NEW WINDOW CALLED "stack119"
549 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 103]
550
551 newstack←FIRST stack IN stacklist WITH stacknumber = 120
552 secrow[ 104]←NEW WINDOW CALLED "stack120"
553 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 104]
554
555 newstack←FIRST stack IN stacklist WITH stacknumber = 121
556 secrow[ 105]←NEW WINDOW CALLED "stack121"
557 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 105]
558
```

```
559 newstack←FIRST stack IN stacklist WITH stacknumber = 122
560 secrow[106]←NEW WINDOW CALLED "stack122"
561 ATTACH HISTORY OF stack_row OF newstack TO secrow[106]
562
563 newstack←FIRST stack IN stacklist WITH stacknumber = 123
564 secrow[107]←NEW WINDOW CALLED "stack123"
565 ATTACH HISTORY OF stack_row OF newstack TO secrow[107]
566
567 newstack←FIRST stack IN stacklist WITH stacknumber = 124
568 secrow[108]←NEW WINDOW CALLED "stack124"
569 ATTACH HISTORY OF stack_row OF newstack TO secrow[108]
570
571 newstack←FIRST stack IN stacklist WITH stacknumber = 125
572 secrow[109]←NEW WINDOW CALLED "stack125"
573 ATTACH HISTORY OF stack_row OF newstack TO secrow[109]
574
575 newstack←FIRST stack IN stacklist WITH stacknumber = 126
576 secrow[110]←NEW WINDOW CALLED "stack126"
577 ATTACH HISTORY OF stack_row OF newstack TO secrow[110]
578
579 newstack←FIRST stack IN stacklist WITH stacknumber = 127
580 secrow[111]←NEW WINDOW CALLED "stack127"
581 ATTACH HISTORY OF stack_row OF newstack TO secrow[111]
582
583 newstack←FIRST stack IN stacklist WITH stacknumber = 128
584 secrow[112]←NEW WINDOW CALLED "stack128"
585 ATTACH HISTORY OF stack_row OF newstack TO secrow[112]
586
587 newstack←FIRST stack IN stacklist WITH stacknumber = 129
588 secrow[113]←NEW WINDOW CALLED "stack129"
589 ATTACH HISTORY OF stack_row OF newstack TO secrow[113]
590
591 newstack←FIRST stack IN stacklist WITH stacknumber = 130
592 secrow[114]←NEW WINDOW CALLED "stack130"
593 ATTACH HISTORY OF stack_row OF newstack TO secrow[114]
594
595 newstack←FIRST stack IN stacklist WITH stacknumber = 131
596 secrow[115]←NEW WINDOW CALLED "stack131"
597 ATTACH HISTORY OF stack_row OF newstack TO secrow[115]
598
599 newstack←FIRST stack IN stacklist WITH stacknumber = 132
600 secrow[116]←NEW WINDOW CALLED "stack132"
601 ATTACH HISTORY OF stack_row OF newstack TO secrow[116]
602
603 newstack←FIRST stack IN stacklist WITH stacknumber = 113
604 secrow[117]←NEW WINDOW CALLED "stack133"
605 ATTACH HISTORY OF stack_row OF newstack TO secrow[117]
606
607 newstack←FIRST stack IN stacklist WITH stacknumber = 134
608 secrow[118]←NEW WINDOW CALLED "stack134"
609 ATTACH HISTORY OF stack_row OF newstack TO secrow[118]
610
611 newstack←FIRST stack IN stacklist WITH stacknumber = 135
612 secrow[119]←NEW WINDOW CALLED "stack135"
613 ATTACH HISTORY OF stack_row OF newstack TO secrow[119]
614
615 newstack←FIRST stack IN stacklist WITH stacknumber = 136
616 secrow[120]←NEW WINDOW CALLED "stack136"
617 ATTACH HISTORY OF stack_row OF newstack TO secrow[120]
618
619 newstack←FIRST stack IN stacklist WITH stacknumber = 137
620 secrow[121]←NEW WINDOW CALLED "stack137"
```

```
621 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 121]
622
623 newstack←FIRST stack IN stacklist WITH stacknumber = 138
624 secrow[ 122]←NEW WINDOW CALLED "stack138"
625 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 122]
626
627 newstack←FIRST stack IN stacklist WITH stacknumber =139
628 secrow[ 123]←NEW WINDOW CALLED "stack139"
629 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 123]
630
631 newstack←FIRST stack IN stacklist WITH stacknumber = 140
632 secrow[ 124]←NEW WINDOW CALLED "stack140"
633 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 124]
634
635 newstack←FIRST stack IN stacklist WITH stacknumber = 141
636 secrow[ 125]←NEW WINDOW CALLED "stack141"
637 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 125]
638
639 newstack←FIRST stack IN stacklist WITH stacknumber = 142
640 secrow[ 126]←NEW WINDOW CALLED "stack142"
641 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 126]
642
643 newstack←FIRST stack IN stacklist WITH stacknumber = 143
644 secrow[ 127]←NEW WINDOW CALLED "stack143"
645 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 127]
646
647 newstack←FIRST stack IN stacklist WITH stacknumber = 144
648 secrow[ 128]←NEW WINDOW CALLED "stack144"
649 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 128]
650
651 newstack←FIRST stack IN stacklist WITH stacknumber = 145
652 secrow[ 129]←NEW WINDOW CALLED "stack145"
653 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 129]
654
655 newstack←FIRST stack IN stacklist WITH stacknumber = 146
656 secrow[ 130]←NEW WINDOW CALLED "stack146"
657 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 130]
658
659 newstack←FIRST stack IN stacklist WITH stacknumber = 147
660 secrow[ 131]←NEW WINDOW CALLED "stack147"
661 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 131]
662
663 newstack←FIRST stack IN stacklist WITH stacknumber = 148
664 secrow[ 132]←NEW WINDOW CALLED "stack148"
665 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 132]
666
667 newstack←FIRST stack IN stacklist WITH stacknumber = 149
668 secrow[ 133]←NEW WINDOW CALLED "stack149"
669 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 133]
670
671 newstack←FIRST stack IN stacklist WITH stacknumber = 150
672 secrow[ 134]←NEW WINDOW CALLED "stack150"
673 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 134]
674
675 newstack←FIRST stack IN stacklist WITH stacknumber = 151
676 secrow[ 135]←NEW WINDOW CALLED "stack151"
677 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 135]
678
679 newstack←FIRST stack IN stacklist WITH stacknumber = 152
680 secrow[ 136]←NEW WINDOW CALLED "stack152"
681 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 136]
682
```

683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710

```
683 newstack←FIRST stack IN stacklist WITH stacknumber = 153
684 secrow[ 137]←NEW WINDOW CALLED "stack153"
685 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 137]
686
687 newstack←FIRST stack IN stacklist WITH stacknumber = 154
688 secrow[ 138]←NEW WINDOW CALLED "stack154"
689 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 138]
690
691 newstack←FIRST stack IN stacklist WITH stacknumber = 155
692 secrow[ 139]←NEW WINDOW CALLED "stack155"
693 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 139]
694
695 newstack←FIRST stack IN stacklist WITH stacknumber = 156
696 secrow[ 140]←NEW WINDOW CALLED "stack156"
697 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 140]
698
699 newstack←FIRST stack IN stacklist WITH stacknumber = 157
700 secrow[ 141]←NEW WINDOW CALLED "stack157"
701 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 141]
702
703 newstack←FIRST stack IN stacklist WITH stacknumber = 158
704 secrow[ 142]←NEW WINDOW CALLED "stack158"
705 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 142]
706
707 newstack←FIRST stack IN stacklist WITH stacknumber =159
708 secrow[ 143]←NEW WINDOW CALLED "stack159"
709 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 143]
710
711 newstack←FIRST stack IN stacklist WITH stacknumber = 160
712 secrow[ 144]←NEW WINDOW CALLED "stack160"
713 ATTACH HISTORY OF stack_row OF newstack TO secrow[ 144]
714 RETURN
```

create portout is a macro that creates portainer outportations

```
 1 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 3
 2 port3queue ← NEW WINDOW CALLED "port3queue"
 3 ATTACH HISTORY OF portlocqueue OF newportloc TO port3queue
 4 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 4
 5 port4queue ← NEW WINDOW CALLED "port4queue"
 6 ATTACH HISTORY OF portlocqueue OF newportloc TO port4queue
 7 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 5
 8 port5queue ← NEW WINDOW CALLED "port5queue"
 9 ATTACH HISTORY OF portlocqueue OF newportloc TO port5queue
10 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 6
11 port6queue ← NEW WINDOW CALLED "port6queue"
12 ATTACH HISTORY OF portlocqueue OF newportloc TO port6queue
13 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 7
14 port7queue ← NEW WINDOW CALLED "port7queue"
15 ATTACH HISTORY OF portlocqueue OF newportloc TO port7queue
16 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 8
17 port8queue ← NEW WINDOW CALLED "port8queue"
18 ATTACH HISTORY OF portlocqueue OF newportloc TO port8queue
19 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 9
20 port9queue ← NEW WINDOW CALLED "port9queue"
21 ATTACH HISTORY OF portlocqueue OF newportloc TO port9queue
22 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 10
23 port10queue ← NEW WINDOW CALLED "port10queue"
24 ATTACH HISTORY OF portlocqueue OF newportloc TO port10queue
25 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 11
26 port11queue ← NEW WINDOW CALLED "port11queue"
27 ATTACH HISTORY OF portlocqueue OF newportloc TO port11queue
28 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 12
29 port12queue ← NEW WINDOW CALLED "port12queue"
30 ATTACH HISTORY OF portlocqueue OF newportloc TO port12queue
31 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 13
32 port13queue ← NEW WINDOW CALLED "port13queue"
33 ATTACH HISTORY OF portlocqueue OF newportloc TO port13queue
34 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 14
35 port14queue ← NEW WINDOW CALLED "port14queue"
36 ATTACH HISTORY OF portlocqueue OF newportloc TO port14queue
37 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 15
38 port15queue ← NEW WINDOW CALLED "port15queue"
39 ATTACH HISTORY OF portlocqueue OF newportloc TO port15queue
40 newportloc ←FIRST portloc IN portlocations WITH portlocnumber = 16
41 port16queue ← NEW WINDOW CALLED "port16queue"
42 ATTACH HISTORY OF portlocqueue OF newportloc TO port16queue
43 RETURN
```

portainertimes is the macro that creates the portainer cycletimes
there are two different cycletimes dependent of the size of the vessel (feeder mega vessel)
And there is a difference between the loading and the unloading of a vessel

qq is the time the portainer eventually has to work

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@Deze formule stelt wel eisen aan het aantal containers per schip hier →
@moet wel rekening mee gehouden worden!!!

@deze formule wordt gebruikt bij het lossen van  de schepen

IF big OF ptship = FALSE & ptship BELONGS TO unloadingship
    n ← LENGTH OF portainerperberth[ shipberthnumber OF ptship]
    PER←ABS(totalunload OF ptship - total OF ptship)
    pers← (total OF ptship/3)
    rep←FLOOR(totalunload OF ptship/pers)

    QQ ← (PER)MODULO(12*n)*(23/((12*n)-1))+ 28.5+(pers-ABS→
    (totalunload OF ptship-pers*rep))*(16/pers)
END
IF BIG OF PTSHIP = TRUE & ptship BELONGS TO unloadingship
    n ← LENGTH OF portainerperberth[ shipberthnumber OF ptship]
    PER←ABS(totalunload OF ptship - total OF ptship)
    pers← (total OF ptship/3)
    rep← FLOOR(totalunload OF ptship/pers)
    QQ ← (PER)MODULO(18*n)*(23/((18*n)-1))+ 28.5+→
    (pers-ABS(totalunload OF ptship-pers*rep))*(16/pers)

END

@deze formule wordt gebruikt bij het laden van de schepen


IF big OF ptship = FALSE & ptship BELONGS TO loadingship
    n ← LENGTH OF portainerperberth[ shipberthnumber OF ptship]
    PER←ABS(totalload OF ptship - totalu OF ptship)
    pers← (totalu OF ptship/3)
    rep←FLOOR(totalload OF ptship/pers)
    QQ ← ((12*n-PER)MODULO(12*n))*(23/((12*n)-1))→
    + 28.5+(totalload OF ptship-pers*rep)*(16/pers)
END

IF BIG OF PTSHIP = TRUE & ptship BELONGS TO loadingship
    n ← LENGTH OF portainerperberth[ shipberthnumber OF ptship]
    PER←ABS(totalload OF ptship - totalu OF ptship)
    pers←(totalu OF ptship/3)
    rep←FLOOR(totalload OF ptship/pers)
    QQ ← (18*N-PER)MODULO(18*n)*(23/((18*n)-1))→
    + 28.5+(totalload OF ptship-pers*rep)*(16/pers)
END
RETURN
```

Appendix 28-IV Clarifying word list

clarifying word list

A **Aantalagv** is an attribute of the crane of the type integer indicating the number of containers that are dropped at a service point by a rgv unloading a vessel, these containers still have to be put into the stack by the stacking crane
y creation is a macro that chooses the cranes that have to serve the stacks. With this macro the number of cranes in the stacks can be varied
acc is the acceleration of the rgv's used during the run
Activeshiplist a set containing the active vessels at the terminal
actlocnr is the actual location number of a portainer
Agv_berth_stack is a macro that creates the route from a berth to the stack
agv_unload creates the route for the rgv to get to the stack from the berth
agv_unload_back is a macro determining the route the rgv has to take from the yard to the berth
agv_stack_berth is a macro that creates the route to get from any stack to a berth
agv_stack_stack is a macro that creates the route from any stack to another stack
agv_uneasy_b_s is a macro that creates a route from the berth to a stack in the transition between loading a vessel and shuffling
Agvberthshufflelist [1 to 3] are 3 sets containing all the rgv's shuffling from that berth
Agvparking a set containing the not active rgv's
agvppactive is a set containing the number of agv's (rgv's) per portainer active
agvppactive_odd is a set containing the number of odd agv's per portainer active
Agvworklist a set containing rgv's with work requests for the trafficmaster
Anchorage is a set containing the vessels waiting to berth
assignedqueue is a set containing rgv's assigned to the crane.

B **Big** is an attribute of ship of the type logical, if big is "false" it means this is a feeder
Bye is an attribute of the rgv of the type logical. If bye of a rgv is true it has to go to the parking

C **claimedlength** is the length the rgv needs to enter the main track
claimset is a set of nodes that have to be claimed in order to enter the main track
control box is the box that appears when starting the run this enables the user to check some graphs after the run
conversation is the macro that creates the control box
crane cycletime is a macro that creates the cycletimes for the stacking cranes
crane outportations is a macro that creates the outportations for the stacking cranes
crane_cycletime is a macro which creates a cycle times for the cranes
Cranelist a set containing all the cranes
cranequeue is a set containing the rgv's waiting to be served by the crane
Create outportations is a macro that creates the outportations
create portout is a macro that creates all the portainer outportations
create_platform is a macro creating the platforms
create_stack is the macro that creates the stacks this macro is called upon by the main
creating sections is a macro that attaches existing queues of sections to windows. In order to monitor these queues. In this way node rows and stack rows can be monitored.

D **distance** is the minimum distance between rgv's when driving on the terminal

E **easy loading** concerns the loading where the containers that have to be loaded are already located in the stacks behind the right berth
Easyload is an attribute of a rgv referring to the easy loading method of a vessel.
end_point is a reference to the last node of a section
Entrance of the rgv is a set containing the starting point of a route of the rgv
Even rgv's take the bottom route when driving at the terminal
exit the first node where the rgv has to leave the main track
exittime is a specific time for a rgv to pull up and leave a node
Exportgen is the exportgenerator and generates the export containers for the vessels

F **firstshuffle** is an attribute of the rgv and indicates this rgv is shuffling for the first time
frombtob[1,2] is an attribute telling how many containers at berth 1 are destined to go to berth 2
full_speed is the top speed of the rgv, this speed can be reached on the main tracks
futlocnr is the future location number of the portainer. The new location of the portainer.

G **Goneshiplist** a set containing the terminated vessels, in this way their attributes can be saved

I **im40** of a vessel is the number of 40 feet import containers
Importgenerator generates the import containers for the vessels
inplacecounter is the counter that counts the containers that are inplace behind that specific berth

K **klaarcount** is the counter that counts containers that are placed on the service points in a stack ready to be taken away by a rgv loading a vessel

L **Loadedagv** is a logical, if false it means the rgv is not carrying a container
Loadingship a set containing all the loading vessels at the terminal
Loadyardallocation is a macro that chooses where the rgv has to get a load container
low_speed is the speed of the rgv in the stacks and at the short cuts

N **n_in** is a reference to a node, the entrance of the stack
nd_x is the x co-ordinate of node
nd_y is the y co-ordinate of a node
node_distance refers to the safety distance at the tracks, this can differ per track
Node_list is a set containing all the nodes
nodespeed id the speed of a rgv when it crosses a node

O **Occupied[1]** is a logical if occupied [1] is true it means berth1 is occupied by a vessel
Odd rgv's take the top route at the terminal when they want to get somewhere at the terminal
Outportation creates the output files
Own stack is a stack behind the portainer the rgv was working for in the beginning

P **pal** is portainer active list
Platforms is a set containing all the platforms
Portaineractivelist a set containing all the active portainers
Portainerperberth [1 to 3] keeps count of the active portainers per berth
portainertimes is the macro that creates the portainer cycletimes
Portasqueue is a set containing rgv's assigned to that portlocation.
portloc is short for portainerlocation
Portlocations a set containing all the portlocations
Portlocqueue is a set containing rgv's waiting under the portainer to be served
Preclaimnodes is a macro called upon by the agv responsible for claiming nodes when more than one node needs to be claimed

Q **quaylocation[l]** is a logical if true it means this location is taken by a portainer
Quayrail a set containing all the portainers

R **Route** of the rgv is a set containing all the exits from the main route the rgv has to take to reach its destination

S **s_length** is the length of a section calculated by the function distance to
sbnr is the stack berth number and identifies behind which berth the stacks are located where the agv is shuffling for if sbnr =0 the rgv is working in the stacks behind its original portlocation (x)
sections is a set containing all the sections
shipanchoragetime is an attribute of the vessel referring to the time in berth
st_x is the actual x co-ordinate of the stack on the terminal
st_y is the actual y co-ordinate of the stack on the terminal
stack row a set containing all the rgv's that are in the stack at that moment
stackcount are the reserved service points in the stack
Stacklist a set containing all the stacks
super_shortcut is a macro designed for an exception at the terminal.

T **tex** is the total of containers to be loaded unto the vessel
Timebig refers to the interarrivaltime of a mega vessel
Timesmall refers to the interarivaltime of a feeder

tmportloc is the portlocation the rgv is working for
tmres40im is the percentage 40 feet importcontainers of all the import containers
tmrestotim is the percentage import containers still to be unloaded of the whole
tmtotalex is the total of containers to be loaded unto the vessel - total already loaded
tmtotalim is the total of containers left on the vessel that have to be unloaded
totalload is the number of containers the vessel has to receive at the terminal here
totalu is the start value of totalload, this value doesn't change in contrast to totalload

U **Unloadinship** a set containing all the vessels that are unloading
unloadyardallocation is a macro that creates the stacks for the rgv's to deliver the unload containers

W **way_in_stack** is a attribute of the stack referring to the way in a stack

X **x** is an attribute of the rgv referring to a portainerlocation
x refers to the current portlocation of the rgv

Y **y** refers to the current stack of the rgv
y_creation is a macro responsible for determining which stacking crane works in which stack
yardshuffleallocation is the macro that creates the locations for the rgv to retrieve containers from when shuffling

Appendix 27-5. Input file for the lay out of the terminal

```
@These sets of 19 represent main tracks, the inter berth lanes @
@node number, x co-ordinate and y co-ordinate @

1       8       194 @in the final version of the program 'eindversie' this maintrack is @
2       70      194 @ not used. @
3       132     194
4       194     194
5       256     194
6       318     194
7       380     194
8       442     194
9       504     194
10      566     194
11      628     194
12      690     194
13      752     194
14      814     194
15      876     194
16      938     194
17      1000    194
18      1062    194
19      1124    194
-1

20

20      1124    186 @in the final version of the program 'eindversie' this maintrack is @
21      1062    186 @ not used. @
22      1000    186
23      938     186
24      876     186
25      814     186
26      752     186
27      690     186
28      628     186
29      566     186
30      504     186
31      442     186
32      380     186
33      318     186
34      256     186
35      194     186
36      132     186
37      70      186
38      8       186
-1

41
41      8       178
42      70      178
43      132     178
44      194     178
45      256     178
46      318     178
47      380     178
48      442     178
```

```
49      504     178
50      566     178
51      628     178
52      690     178
53      752     178
54      814     178
55      876     178
56      938     178
57      1000    178
58      1062    178
59      1124    178
-1

60
60      1124    170
61      1062    170
62      1000    170
63      938     170
64      876     170
65      814     170
66      752     170
67      690     170
68      628     170
69      566     170
70      504     170
71      442     170
72      380     170
73      318     170
74      256     170
75      194     170
76      132     170
77      70      170
78      8       170
-1
81
81      8       162
82      70      162
83      132     162
84      194     162
85      256     162
86      318     162
87      380     162
88      442     162
89      504     162
90      566     162
91      628     162
92      690     162
93      752     162
94      814     162
95      876     162
96      938     162
97      1000    162
98      1062    162
99      1124    162
-1
100

100     1124    154
101     1062    154
102     1000    154
```

```
104     876     154
105     814     154
106     752     154
107     690     154
108     628     154
109     566     154
110     504     154
111     442     154
112     380     154
113     318     154
114     256     154
115     194     154
116     132     154
117     70      154
118     8       154
-1

501
501     8       42
502     70      42
503     132     42
504     194     42
505     256     42
506     318     42
507     380     42
508     442     42
509     504     42
510     566     42
511     628     42
512     690     42
513     752     42
514     814     42
515     876     42
516     938     42
517     1000    42
518     1062    42
519     1124    42
-1
520
520     1124    34
521     1062    34
522     1000    34
523     938     34
524     876     34
525     814     34
526     752     34
527     690     34
528     628     34
529     566     34
530     504     34
531     442     34
532     380     34
533     318     34
534     256     34
535     194     34
536     132     34
537     70      34
538     8       34
-1
541
```

```
541

541     8       26
542     70      26
543     132     26
544     194     26
545     256     26
546     318     26
547     380     26
548     442     26
549     504     26
550     566     26
551     628     26
552     690     26
553     752     26
554     814     26
555     876     26
556     938     26
557     1000    26
558     1062    26
559     1124    26
-1
560
560     1124    18
561     1062    18
562     1000    18
563     938     18
564     876     18
565     814     18
566     752     18
567     690     18
568     628     18
569     566     18
570     504     18
571     442     18
572     380     18
573     318     18
574     256     18
575     194     18
576     132     18
577     70      18
578     8       18
-1
581
581     8       10 @in the final version of the program 'eindversie' this maintrack is @
582     70      10 @not used@
583     132     10
584     194     10
585     256     10
586     318     10
587     380     10
588     442     10
589     504     10
590     566     10
591     628     10
592     690     10
593     752     10
594     814     10
595     876     10
596     938     10
```

```
598     1062    10
599     1124    10
-1

600
600     1124    2 @in the final version of the program 'eindversie' this maintrack is @
601     1062    2 @ not used. @
602     1000    2
603     938     2
604     876     2
605     814     2
606     752     2
607     690     2
608     628     2
609     566     2
610     504     2
611     442     2
612     380     2
613     318     2
614     256     2
615     194     2
616     132     2
617     70      2
618     8       2
-1

@These sets of 13 represent main tracks, the stack lanes @
@nodenumber, x co-ordinate and y co-ordinate @
201
201     8       146 @in the final version of the program 'eindversie' this stacklane is @
202     8       134 @not used @
203     8       131
204     8       119
205     8       116
206     8       104
207     8       101
208     8       89
209     8       86
210     8       74
211     8       66
212     8       58
213     8       50
-1
218
218     70      50 @in the final version of the program 'eindversie' this stacklane is @
219     70      58 @not used @
220     70      66
221     70      74
222     70      86
223     70      89
224     70      101
225     70      104
226     70      116
227     70      119
228     70      131
229     70      134
230     70      146
-1
231
231     132     146
```

```
231     132     146
232     132     134
233     132     131
234     132     119
235     132     116
236     132     104
237     132     101
238     132     89
239     132     86
240     132     74
241     132     66
242     132     58
243     132     50
-1
248
248     194     50
249     194     58
250     194     66
251     194     74
252     194     86
253     194     89
254     194     101
255     194     104
256     194     116
257     194     119
258     194     131
259     194     134
260     194     146
-1
261
261     256     146
262     256     134
263     256     131
264     256     119
265     256     116
266     256     104
267     256     101
268     256     89
269     256     86
270     256     74
271     256     66
272     256     58
273     256     50
-1
278
278     318     50
279     318     58
280     318     66
281     318     74
282     318     86
283     318     89
284     318     101
285     318     104
286     318     116
287     318     119
288     318     131
289     318     134
290     318     146
-1
291
```

292 381 134
293 381 131
294 381 119
295 381 116
296 381 104
297 381 101
298 381 89
299 381 86
300 381 74
301 381 66
302 381 58
303 381 50
-1
308
308 442 50
309 442 58
310 442 66
311 442 74
312 442 86
313 442 89
314 442 101
315 442 104
316 442 116
317 442 119
318 442 131
319 442 134
320 442 146
-1
321
321 504 146
322 504 134
323 504 131
324 504 119
325 504 116
326 504 104
327 504 101
328 504 89
329 504 86
330 504 74
331 504 66
332 504 58
333 504 50
-1
338
338 566 50
339 566 58
340 566 66
341 566 74
342 566 86
343 566 89
344 566 101
345 566 104
346 566 116
347 566 119
348 566 131
349 566 134
350 566 146
-1
351
351 628 146

```
351     628     146
352     628     134
353     628     131
354     628     119
355     628     116
356     628     104
357     628     101
358     628     89
359     628     86
360     628     74
361     628     66
362     628     58
363     628     50
-1
368
368     692     50
369     692     58
370     692     66
371     692     74
372     692     86
373     692     89
374     692     101
375     692     104
376     692     116
377     692     119
378     692     131
379     692     134
380     692     146
-1
381
381     752     146
382     752     134
383     752     131
384     752     119
385     752     116
386     752     104
387     752     101
388     752     89
389     752     86
390     752     74
391     752     66
392     752     58
393     752     50
-1
398
398     814     50
399     814     58
400     814     66
401     814     74
402     814     86
403     814     89
404     814     101
405     814     104
406     814     116
407     814     119
408     814     131
409     814     134
410     814     146
-1
411
```

```
412     876     134
413     876     131
414     876     119
415     876     116
416     876     104
417     876     101
418     876     89
419     876     86
420     876     74
421     876     66
422     876     58
423     876     50
-1
428
428     938     50
429     938     58
430     938     66
431     938     74
432     938     86
433     938     89
434     938     101
435     938     104
436     938     116
437     938     119
438     938     131
439     938     134
440     938     146
-1
441
441     1000    146
442     1000    134
443     1000    131
444     1000    119
445     1000    116
446     1000    104
447     1000    101
448     1000    89
449     1000    86
450     1000    74
451     1000    66
452     1000    58
453     1000    50
-1
458
458     1062    50 @in the final version of the program 'eindversie' this stacklane is @
459     1062    58 @not used@
460     1062    66
461     1062    74
462     1062    86
463     1062    89
464     1062    101
465     1062    104
466     1062    116
467     1062    119
468     1062    131
469     1062    134
470     1062    146
-1
471
471     1124    146 @in the final version of the program 'eindversie' this stacklane is @
```

```
471     1124    146 @in the final version of the program 'eindversie' this stacklane is @
472     1124    134 @not used@
473     1124    131
474     1124    119
475     1124    116
476     1124    104
477     1124    101
478     1124    89
479     1124    86
480     1124    74
481     1124    66
482     1124    58
483     1124    50
-1
-1

@stacks@
@name, node_in, node_out, x co-ordinate and y co-ordinate@
stack1          201     230     39      146 @in the final version of the program @
stack2          229     202     39      134 @ 'eindversie' these stacks are not used @
stack3          203     228     39      131
stack4          227     204     39      119
stack5          205     226     39      116
stack6          225     206     39      104
stack7          207     224     39      101
stack8          223     208     39      89
stack9          209     222     39      86
stack10         221     210     39      74

stack11         230     231     101     146 @in the final version of the program @
stack12         232     229     101     134 @ 'eindversie' these stacks are not used @
stack13         228     233     101     131
stack14         234     227     101     119
stack15         226     235     101     116
stack16         236     225     101     104
stack17         224     237     101     101
stack18         238     223     101     89
stack19         222     239     101     86
stack20         240     221     101     74

stack21         231     260     163     146
stack22         259     232     163     134
stack23         233     258     163     131
stack24         257     234     163     119
stack25         235     256     163     116
stack26         255     236     163     104
stack27         237     254     163     101
stack28         253     238     163     89
stack29         239     252     163     86
stack30         251     240     163     74

stack31         260     261     225     146
stack32         262     259     225     134
stack33         258     263     225     131
stack34         264     257     225     119
stack35         256     265     225     116
stack36         266     255     225     104
stack37         254     267     225     101
stack38         268     253     225     89
stack39         252     269     225     86
```

stack41	261	290	287	146
stack42	289	262	287	134
stack43	263	288	287	131
stack44	287	264	287	119
stack45	265	286	287	116
stack46	285	266	287	104
stack47	267	284	287	101
stack48	283	268	287	89
stack49	269	282	287	86
stack50	281	270	287	74
stack51	290	291	349	146
stack52	292	289	349	134
stack53	288	293	349	131
stack54	294	287	349	119
stack55	286	295	349	116
stack56	296	285	349	104
stack57	284	297	349	101
stack58	298	283	349	89
stack59	282	299	349	86
stack60	300	281	349	74
stack61	291	320	411	146
stack62	319	292	411	134
stack63	293	318	411	131
stack64	317	294	411	119
stack65	295	316	411	116
stack66	315	296	411	104
stack67	297	314	411	101
stack68	313	298	411	89
stack69	299	312	411	86
stack70	311	300	411	74
stack71	320	321	473	146
stack72	322	319	473	134
stack73	318	323	473	131
stack74	324	317	473	119
stack75	316	325	473	116
stack76	326	315	473	104
stack77	314	327	473	101
stack78	328	313	473	89
stack79	312	329	473	86
stack80	330	311	473	74
stack81	321	350	535	146
stack82	349	322	535	134
stack83	323	348	535	131
stack84	347	324	535	119
stack85	325	346	535	116
stack86	345	326	535	104
stack87	327	344	535	101
stack88	343	328	535	89
stack89	329	342	535	86
stack90	341	330	535	74
stack91	350	351	597	146
stack92	352	349	597	134
stack93	348	353	597	131
stack94	354	347	597	119

stack94	354	347	597	119
stack95	346	355	597	116
stack96	356	345	597	104
stack97	344	357	597	101
stack98	358	343	597	89
stack99	342	359	597	86
stack100	360	341	597	74
stack101	351	380	659	146
stack102	379	352	659	134
stack103	353	378	659	131
stack104	377	354	659	119
stack105	355	376	659	116
stack106	375	356	659	104
stack107	357	374	659	101
stack108	373	358	659	89
stack109	359	372	659	86
stack110	371	360	659	74
stack111	380	381	721	146
stack112	382	379	721	134
stack113	378	383	721	131
stack114	384	377	721	119
stack115	376	385	721	116
stack116	386	375	721	104
stack117	374	387	721	101
stack118	388	373	721	89
stack119	372	389	721	86
stack120	390	371	721	74
stack121	381	410	783	146
stack122	409	382	783	134
stack123	383	408	783	131
stack124	407	384	783	119
stack125	385	406	783	116
stack126	405	386	783	104
stack127	387	404	783	101
stack128	403	388	783	89
stack129	389	402	783	86
stack130	401	390	783	74
stack131	410	411	845	146
stack132	412	409	845	134
stack133	408	413	845	131
stack134	414	407	845	119
stack135	406	415	845	116
stack136	416	405	845	104
stack137	404	417	845	101
stack138	418	403	845	89
stack139	402	419	845	86
stack140	420	401	845	74
stack141	411	440	907	146
stack142	439	412	907	134
stack143	413	438	907	131
stack144	437	414	907	119
stack145	415	436	907	116
stack146	435	416	907	104
stack147	417	434	907	101
stack148	433	418	907	89

```
stack150    431    420    907    74

stack151    440    441    969    146
stack152    442    439    969    134
stack153    438    443    969    131
stack154    444    437    969    119
stack155    436    445    969    116
stack156    446    435    969    104
stack157    434    447    969    101
stack158    448    433    969    89
stack159    432    449    969    86
stack160    450    431    969    74

stack161    441    470    1031   146 @in the final version of the program@
stack162    469    442    1031   134 @ 'eindversie' these stacks are not used @
stack163    443    468    1031   131
stack164    467    444    1031   119
stack165    445    466    1031   116
stack166    465    446    1031   104
stack167    447    464    1031   101
stack168    463    448    1031   89
stack169    449    462    1031   86
stack170    461    450    1031   74

stack171    470    471    1093   146  @in the final version of the program@
stack172    472    469    1093   134  @ 'eindversie' these stacks are not used @
stack173    468    473    1093   131
stack174    474    467    1093   119
stack175    466    475    1093   116
stack176    476    465    1093   104
stack177    464    477    1093   101
stack178    478    463    1093   89
stack179    462    479    1093   86
stack180    480    461    1093   74


END

@portainers@
@name, node_in, node_out, x co-ordinate and y co-ordinate@

loc1los 212    219    39     58          @in the final version of the program@
loc1laad       218    213    39     50   @ 'eindversie' these are not used @

loc2los 242    219    101    58          @in the final version of the program@
@loc2laad      218    243    101    50   @ 'eindversie' these are not used @

loc3los 242    249    163    58
loc3laad       248    243    163    50

loc4los 272    249    225    58
@loc4laad      248    273    225    50@

loc5los 272    279    287    58
loc5laad       278    273    287    50

loc6los 302    279    349    58
@loc6laad      278    303    349    50@

loc7los 302    309    411    58
```

```
loc7los 302     309     411     58
loc7laad        308     303     411     50

loc8los 332     309     473     58
@loc8laad       308     333     473     50@

loc9los 332     339     535     58
loc9laad        338     333     535     50

loc10los        362     339     597     58
@loc10laad      338     363     597     50@

loc11los        362     369     659     58
loc11laad       368     363     659     50

loc12los        392     369     721     58
@loc12laad      368     393     721     50@

loc13los        392     399     783     58
loc13laad       398     393     783     50

loc14los        422     399     845     58
@loc14laad      398     423     845     50@

loc15los        422     429     907     58
loc15laad       428     423     907     50

loc16los        452     429     969     58
@loc16laad      428     453     969     50@

loc17los        452     459     1031    58
loc17laad       458     453     1031    50

loc18los        482     459     1093    58
@loc18laad      458     483     1093    50@

END

@short cuts@
@entrance_node of the short cut, exit_node of the short cut@
211     220

241     250

271     280

301     310

331     340

361     370

391     400

421     430

451     460


38      41
```

78 81
81 118
118 201

230 117
117 82
82 77
77 42
42 37
37 2

3 36
36 43
43 76
76 83
83 116
116 231

260 115
115 84
84 75
75 44
44 35
35 4

5 34
34 45
45 74
74 85
85 114
114 261

290 113
113 86
86 73
73 46
46 33
33 6

7 32
32 47
47 72
72 87
87 112
112 291

320 111
111 88
88 71
71 48
48 31
31 8

9 30
30 49
49 70
70 89
89 110
110 321

350 109
109 90
90 69
69 50
50 29
29 10

11 28
28 51
51 68
68 91
91 108
108 351

380 107
107 92
92 67
67 52
52 27
27 12

13 26
26 53
53 66
66 93
93 106
106 381

410 105
105 94
94 65
65 54
54 25
25 14

15 24
24 55
55 64
64 95
95 104
104 411

440 103
103 96
96 63
63 56
56 23
23 16

17 22
22 57
57 62
62 97
97 102
102 441

470 101
101 98
98 61

58 21
21 18

19 20
20 59
59 60
60 99
99 100
100 471

213 501
501 538
538 541
541 578
578 581
581 618

617 582
582 577
577 542
542 537
537 502
502 218

243 503
503 536
536 543
543 576
576 583
583 616

615 584
584 575
575 544
544 535
535 504
504 248

273 505
505 534
534 545
545 574
574 585
585 614

613 586
586 573
573 546
546 533
533 506
506 278

303 507
507 532
532 547
547 572
572 587
587 612

611 588

611 588
588 571
571 548
548 531
531 508
508 308

333 509
509 530
530 549
549 570
570 589
589 610

609 590
590 569
569 550
550 529
529 510
510 338

363 511
511 528
528 551
551 568
568 591
591 608

607 592
592 567
567 552
552 527
527 512
512 368

393 513
513 526
526 553
553 566
566 593
593 606

605 594
594 565
565 554
554 525
525 514
514 398

423 515
515 524
524 555
555 564
564 595
595 604

603 596
596 563
563 556
556 523

```
516     428

453     517
517     522
522     557
557     562
562     597
597     602

601     598
598     561
561     558
558     521
521     518
518     458

483     519
519     520
520     559
559     560
560     599
599     600
481     460

-1

@super shortcuts@
242     219
272     249
302     279
332     309
362     339
392     369
422     399
452     429
482     459

241     220
271     250
301     280
331     310
361     340
391     370
421     400
451     430
481     460

230     231
260     261
290     291
320     321
350     351
380     381
410     411
440     441
470     471

218     243
248     273
278     303
```

278 303
308 333
338 363
368 393
398 423
428 453
458 483

-1
201
201

Appendix 28-5. Input file for the lay out of the terminal

```
@These sets of 19 represent main tracks, the inter berth lanes @
@node number, x co-ordinate and y co-ordinate @

1       8       194 @in the final version of the program 'eindversie' this maintrack is @
2       70      194 @ not used. @
3       132     194
4       194     194
5       256     194
6       318     194
7       380     194
8       442     194
9       504     194
10      566     194
11      628     194
12      690     194
13      752     194
14      814     194
15      876     194
16      938     194
17      1000    194
18      1062    194
19      1124    194
-1

20

20      1124    186 @in the final version of the program 'eindversie' this maintrack is @
21      1062    186 @ not used. @
22      1000    186
23      938     186
24      876     186
25      814     186
26      752     186
27      690     186
28      628     186
29      566     186
30      504     186
31      442     186
32      380     186
33      318     186
34      256     186
35      194     186
36      132     186
37      70      186
38      8       186
-1

41
41      8       178
42      70      178
43      132     178
44      194     178
45      256     178
46      318     178
47      380     178
48      442     178
```

```
49      504     178
50      566     178
51      628     178
52      690     178
53      752     178
54      814     178
55      876     178
56      938     178
57      1000    178
58      1062    178
59      1124    178
-1

60
60      1124    170
61      1062    170
62      1000    170
63      938     170
64      876     170
65      814     170
66      752     170
67      690     170
68      628     170
69      566     170
70      504     170
71      442     170
72      380     170
73      318     170
74      256     170
75      194     170
76      132     170
77      70      170
78      8       170
-1
81
81      8       162
82      70      162
83      132     162
84      194     162
85      256     162
86      318     162
87      380     162
88      442     162
89      504     162
90      566     162
91      628     162
92      690     162
93      752     162
94      814     162
95      876     162
96      938     162
97      1000    162
98      1062    162
99      1124    162
-1
100

100     1124    154
101     1062    154
102     1000    154
```

Appendix 28-VI Input file Rgv attributes

Using an input file the attributes of the rgv can be changed. The listing of the input file is shown below. This input file is called trolley_conf.txt.

@configuration in file trolley_conf.txt@

@Rgv speed in m/s on main route @ 3
@ Rgv speed in m/s passing stack lane @ 1
@Rgv speed in m/s passing short cut@ 2

@handling time in seconds to pick up or deliver a load@ 30
@minimum distance between trolleys on the main track@ 3